



# **KN210 CPU Module Set System Maintenance**

**Order Number EK-329AB-MG-002**

**digital equipment corporation  
maynard, massachusetts**

---

**First Printing, July 1989**

**Revised, December 1989**

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation.

Digital Equipment Corporation assumes no responsibility for any errors that may appear in this document.

Any software described in this document is furnished under a license and may be used or copied only in accordance with the terms of such license. No responsibility is assumed for the use or reliability of software or equipment that is not supplied by Digital Equipment Corporation or its affiliated companies.


**Restricted Rights:** Use, duplication, or disclosure by the U.S. Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

© Digital Equipment Corporation 1989

All rights reserved. Printed in U.S.A.

The Reader's Comments form at the end of this document requests your critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

COMPACTape	DIBOL	RT
DDCMP	DSSI	ThinWire
DEC	MASSBUS	ULTRIX
DECmate	MicroVAX	UNIBUS
DECnet	PDP	VAX
DECserver	P/OS	VAXcluster
DECsystem 5400	Professional	VAX DOCUMENT
DECUS	Q-bus	VAXELN
DECwriter	Rainbow	VAXlab
DELNI	ReGIS	VMS
DELQA	RQDX	VT
DEQNA	RSTS	Work Processor
DESTA	RSX	

**FCC NOTICE:** The equipment described in this manual generates, uses, and may emit radio frequency energy. The equipment has been type tested and found to comply with the limits for a Class A computing device pursuant to Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such radio frequency interference when operated in a commercial environment. Operation of this equipment in a residential area may cause interference, in which case the user at his own expense may be required to take measures to correct the interference.

**S1329**

This document was prepared using VAX DOCUMENT, Version 1.2.

# Contents

---

## Preface

xi

## Chapter 1 KN210 CPU and Memory Subsystem

---

1.1	KN210 Overview .....	1-1
1.2	KN210 Features .....	1-7
1.2.1	R3000 RISC Processor .....	1-8
1.2.2	Floating-Point Accelerator .....	1-8
1.2.3	Cache Memory .....	1-8
1.2.4	Memory Controller .....	1-9
1.2.5	Console Serial Line .....	1-9
1.2.6	Time-of-Year Clock and Timers .....	1-9
1.2.7	Boot and Diagnostic Facility .....	1-9
1.2.8	Q22-bus Interface .....	1-10
1.2.9	CVAX Diagnostic Processor .....	1-10
1.2.10	Network Interface .....	1-11
1.2.11	Mass Storage Interface .....	1-11
1.3	H3602-AB CPU I/O Panel .....	1-12
1.4	MS650-BA Memory .....	1-15

## Chapter 2 Configuration

---

2.1	Introduction .....	2-1
2.2	General Module Order .....	2-1
2.3	Module Order for KN210 Systems .....	2-2
2.4	Module Configuration .....	2-2
2.5	DSSI Configuration .....	2-4
2.5.1	Changing the Node Name .....	2-5
2.5.2	Changing the Unit Number .....	2-6
2.5.3	DSSI Cabling .....	2-8



2.6	Configuration Worksheet .....	2-11
-----	-------------------------------	------

## Chapter 3 KN210 Firmware

---

3.1	Introduction .....	3-1
3.2	KN210 Firmware Features .....	3-2
3.3	CVAX Halt Entry and Dispatch Code .....	3-3
3.4	Power-Up .....	3-5
3.4.1	Power-Up Sequence: Operation Switch Set to Normal! ....	3-5
3.4.2	Power-Up Sequence: Operation Switch Set to Maintenance	3-6
3.4.3	Operation Switch Set to Action: Loopback Tests .....	3-6
3.4.4	Operation Switch Set to Action: Language Query .....	3-7
3.5	Bootstrap .....	3-8
3.5.1	ULTRIX-32 Bootstrap .....	3-9
3.5.2	MDM Bootstrap .....	3-10
3.5.3	MDM Restart .....	3-14
3.6	Normal Mode Overview .....	3-15
3.6.1	Control Characters in Normal Mode .....	3-15
3.6.2	Environment Variables in Normal Mode .....	3-16
3.7	Normal Mode Commands .....	3-17
3.7.1	Conventions Used in This Section .....	3-19
3.7.2	Getting Help .....	3-19
3.7.3	boot .....	3-20
3.7.4	continue .....	3-23
3.7.5	d (deposit) .....	3-24
3.7.6	dump .....	3-25
3.7.7	e (examine) .....	3-27
3.7.8	fill .....	3-28
3.7.9	go .....	3-29
3.7.10	help .....	3-30
3.7.11	? .....	3-31
3.7.12	init .....	3-32
3.7.13	printenv .....	3-33
3.7.14	setenv .....	3-34
3.7.15	unsetenv .....	3-35
3.8	Maintenance Mode Overview .....	3-36

3.8.1	Command Syntax in Maintenance Mode . . . . .	3-37
3.8.2	Address Specifiers in Maintenance Mode . . . . .	3-37
3.8.3	Symbolic Addresses in Maintenance Mode . . . . .	3-38
3.8.4	Command Qualifiers in Maintenance Mode . . . . .	3-41
3.8.5	Maintenance Mode Command Keywords . . . . .	3-43
3.9	Maintenance Mode Commands . . . . .	3-45
3.9.1	BOOT . . . . .	3-45
3.9.2	CONFIGURE . . . . .	3-46
3.9.3	CONTINUE . . . . .	3-48
3.9.4	DEPOSIT . . . . .	3-49
3.9.5	EXAMINE . . . . .	3-50
3.9.6	EXIT . . . . .	3-52
3.9.7	FIND . . . . .	3-53
3.9.8	HALT . . . . .	3-54
3.9.9	HELP . . . . .	3-55
3.9.10	INITIALIZE . . . . .	3-57
3.9.11	MOVE . . . . .	3-59
3.9.12	NEXT . . . . .	3-61
3.9.13	REPEAT . . . . .	3-63
3.9.14	SEARCH . . . . .	3-64
3.9.15	SET . . . . .	3-66
3.9.16	SHOW . . . . .	3-69
3.9.17	START . . . . .	3-73
3.9.18	TEST . . . . .	3-74
3.9.19	UNJAM . . . . .	3-75
3.9.20	X—Binary Load and Unload . . . . .	3-76
3.9.21	! (Comment) . . . . .	3-78

## **Chapter 4 Troubleshooting and Diagnostics**

---

4.1	Introduction . . . . .	4-1
4.2	General Procedures . . . . .	4-1
4.3	KN210 CVAX ROM-Based Diagnostics . . . . .	4-3
4.3.1	Diagnostic Tests . . . . .	4-4
4.3.2	Scripts . . . . .	4-7
4.3.3	Script Calling Sequence . . . . .	4-9

4.3.4	Creating Scripts .....	4-11
4.3.5	Console Displays .....	4-15
4.3.6	System Halt Messages .....	4-27
4.3.7	Console Error Messages .....	4-28
4.3.8	VMB Error Messages (CVAX) .....	4-29
4.4	Acceptance Testing .....	4-30
4.5	Troubleshooting .....	4-37
4.5.1	FE Utility .....	4-37
4.5.2	Isolating Memory Failures .....	4-40
4.5.3	Additional Troubleshooting Suggestions .....	4-43
4.6	Loopback Tests .....	4-44
4.6.1	DSSI Problems .....	4-44
4.6.2	Ethernet Problems .....	4-44
4.6.3	Testing the Console Port .....	4-45
4.7	Module Self-Tests .....	4-45
4.8	RF-Series ISE Troubleshooting and Diagnostics .....	4-46
4.8.1	DRVSTST .....	4-48
4.8.2	DRVEXR .....	4-49
4.8.3	HISTORY .....	4-50
4.8.4	ERASE .....	4-51
4.8.5	PARAMS .....	4-52
4.8.5.1	EXIT .....	4-53
4.8.5.2	HELP .....	4-53
4.8.5.3	SET .....	4-53
4.8.5.4	SHOW .....	4-54
4.8.5.5	STATUS .....	4-54
4.8.5.6	WRITE .....	4-54
4.8.6	Diagnostic Error Codes .....	4-55

## Appendix A ULTRIX-32 Exerciser and Uerf Command Summary

---

A.1	Online ULTRIX Exerciser .....	A-1
A.1.1	Communications Exerciser (Asynchronous Serial Lines) ..	A-2
A.1.2	Disk Exerciser .....	A-3
A.1.3	File System Exerciser .....	A-4

A.1.4	Line Printer Exerciser .....	A-4
A.1.5	Memory Exerciser .....	A-5
A.1.6	Magtape Exerciser .....	A-6
A.1.7	TCP/IP Network Exerciser .....	A-7
A.2	User Error Log Commands .....	A-8

## **Appendix B KN210 Address Assignments**

---

B.1	Accessing Physical Locations (R3000) .....	B-1
B.2	KN210 CPU Module .....	B-3
B.2.1	R3000 Physical Address Space Map (M7635-AA) .....	B-3
B.2.2	KN210 Diagnostic Processor Physical Address Space Map (M7635-AA) .....	B-6
B.2.3	Diagnostic Processor Registers .....	B-8
B.2.4	Global Q22-bus Memory Space Map .....	B-10
B.3	KN210 I/O Module .....	B-11
B.3.1	R3000 Physical I/O Address Space Map (M7636-AA) ....	B-11
B.3.2	Diagnostic Physical I/O Address Space Map (M7636-AA) .	B-12

## **Appendix C Configuring the KFQSA**

---

C.1	KFQSA Overview .....	C-1
C.2	Configuring the KFQSA at Installation .....	C-2
C.2.1	Entering Maintenance Mode .....	C-4
C.2.2	Displaying Current Addresses .....	C-5
C.2.3	Running the Configure Utility .....	C-6
C.3	Programming the KFQSA .....	C-8
C.4	Reprogramming the KFQSA .....	C-13
C.5	Changing the ISE Unit Number .....	C-15

## **Appendix D Field Replaceable Units (FRUs)**

---

## Appendix E Related Documentation

---

### Index

---

#### Examples

---

2-1	Changing a DSSI Node Name .....	2-5
2-2	Changing a DSSI Unit Number .....	2-7
3-1	Language Selection Menu .....	3-8
3-2	Command Procedure to Boot on Installation .....	3-10
4-1	Creating a Script with Utility 9F .....	4-12
4-2	Listing and Repeating Tests with Utility 9F .....	4-14
4-3	Console Display (No Errors) .....	4-15
4-4	Sample Output with Errors (CVAX) .....	4-16
4-5	Sample Output with Errors (R3000) .....	4-16
4-6	FE Utility Example .....	4-37
4-7	Isolating Bad Memory Using T 9C .....	4-42
4-8	9C—Conditions for Determining a Memory FRU .....	4-43
C-1	KFQSA (M7769) Service Mode Switch Settings .....	C-4
C-2	Entering Console Mode Display .....	C-5
C-3	SHOW QBUS Display .....	C-5
C-4	Configure Display .....	C-7
C-5	Display for Programming the First KFQSA .....	C-9
C-6	SHOW QBUS Display .....	C-11
C-7	SHOW DEVICE Display .....	C-12
C-8	Reprogramming the KFQSA Display .....	C-14
C-9	Display for Changing Unit Number .....	C-16

#### Figures

---

1-1	KN210 CPU Module (M7635-AA) .....	1-2
1-2	KN210 I/O Module (M7636-AA) .....	1-3
1-3	KN210 CPU Module Set Functional Block Diagram .....	1-4
1-4	Location of KN210 and Memory Modules in BA213 Card Cage .....	1-6
1-5	H3602-AB CPU I/O Panel .....	1-13

1-6	MS650-BA Memory Module (M7622-AA) . . . . .	1-16
2-1	DSSI Cabling, BA213 Enclosure . . . . .	2-9
2-2	RF-Series OCP . . . . .	2-10
2-3	BA213 Configuration Worksheet . . . . .	2-13
4-1	KN210 CPU Module LEDs . . . . .	4-19
B-1	KN210 Virtual Memory Map . . . . .	B-2
C-1	KFQSA Module Layout (M7769) . . . . .	C-3

## Tables

1-1	H3602-AB Operation and Function Switch Settings . . . . .	1-14
2-1	ISE Order in BA213 Enclosure . . . . .	2-4
2-2	RF71 DIP Switch Settings . . . . .	2-4
2-3	Power and Bus Loads for KN210 Options . . . . .	2-12
3-1	KN210 Firmware Code . . . . .	3-1
3-2	Actions Taken on a Halt . . . . .	3-4
3-3	ULTRIX-32 Supported Boot Devices . . . . .	3-9
3-4	VMC Boot Flags . . . . .	3-12
3-5	Supported MDM Boot Devices . . . . .	3-13
3-6	Normal-Mode Control Characters . . . . .	3-15
3-7	Environment Variables . . . . .	3-16
3-8	KN210 Normal Mode Commands . . . . .	3-17
3-9	Boot Device Names (Normal Mode) . . . . .	3-20
3-10	KN210 Console Control Characters (Maintenance Mode) . . . . .	3-36
3-11	Console Symbolic Addresses (Maintenance Mode) . . . . .	3-38
3-12	Symbolic Addresses Used in Any Address Space . . . . .	3-41
3-13	Console Command Qualifiers (Maintenance Mode) . . . . .	3-42
3-14	Command Keywords by Type (Maintenance Mode) . . . . .	3-43
3-15	Console Command Summary (Maintenance Mode) . . . . .	3-43
4-1	Test and Utility Numbers . . . . .	4-5
4-2	Scripts Available to Customer Services . . . . .	4-8
4-3	Commonly Used Field Service Scripts . . . . .	4-9
4-4	Tests Run During Power-Up . . . . .	4-15
4-5	Values Saved, Machine Check Exception During Executive (CVAX) . . . . .	4-18
4-6	Values Saved, Exception During Executive (CVAX) . . . . .	4-18
4-7	KN210 Console Displays and FRUs . . . . .	4-20

4-8	System Halt Messages .....	4-27
4-9	Console Error Messages .....	4-28
4-10	VMB Error Messages .....	4-29
4-11	Hardware Error Summary Register .....	4-39
4-12	Loopback Connectors for Q22-bus Devices .....	4-46
4-13	DRVST Messages .....	4-48
4-14	DRVEXR Messages .....	4-49
4-15	HISTORY Messages .....	4-50
4-16	ERASE Messages .....	4-52
4-17	RF-Series ISE Diagnostic Error Codes .....	4-55
B-1	R3000 Memory Space .....	B-3
B-2	R3000 Input/Output Space .....	B-3
B-3	Diagnostic Processor Memory Space .....	B-6
B-4	Diagnostic Processor Input/Output Space .....	B-6
B-5	Diagnostic Processor Registers .....	B-8
B-6	Q22-bus Memory Space .....	B-10
B-7	Q22-bus I/O Space with BBS7 Asserted .....	B-10
B-8	R3000 Physical Addresses .....	B-11
B-9	Diagnostic Input/Output Addresses .....	B-12

# Preface

---

This guide describes the base system, configuration, ROM-based diagnostics, and troubleshooting procedures for systems containing the KN210 CPU module and the KN210 I/O module.

## Intended Audience

This guide is intended for use by Digital Customer Services personnel and qualified self-maintenance customers.

## Organization

This guide has four chapters and five appendixes, as follows:

Chapter 1 describes the KN210 CPU module set and the MS650-BA memory.

Chapter 2 contains system configuration guidelines and provides a table listing current, power, and bus loads for supported options. It also describes the Digital Storage System Interconnect (DSSI) bus interface cabling between the KN210 I/O module, the CPU I/O panel, the operator control panel (OCP), and the RF-series integrated storage elements (ISEs).

Chapter 3 describes the KN210 diagnostic firmware, including maintenance mode commands and normal mode commands.

Chapter 4 describes the KN210 diagnostics, including an error message and FRU cross-reference table. It also describes diagnostics that reside on the RF-series ISEs.

Appendix A contains a summary of ULTRIX-32 Exerciser and Uerf commands.

Appendix B lists the address space for the KN210 CPU module and the KN210 I/O module.

Appendix C explains how to configure the KFQSA storage adapter.

Appendix D lists the major field replaceable units (FRUs) associated with the KN210 system.

Appendix E contains a list of related documentation.



## **Cautions, Differences, and Notes**

Cautions, differences, and notes appear throughout this guide. They have the following meanings:

<b>CAUTION</b>	Provides information to prevent damage to equipment or software.
<b>DIFFERENCES</b>	Provides information on key differences between the KN210 system (DECsystem 5400) and MicroVAX systems that are similar.
<b>NOTE</b>	Provides general information about the current topic.



# KN210 CPU and Memory Subsystem

---

This chapter outlines the KN210 CPU module, the KN210 I/O module, and the MS650-BA memory module.

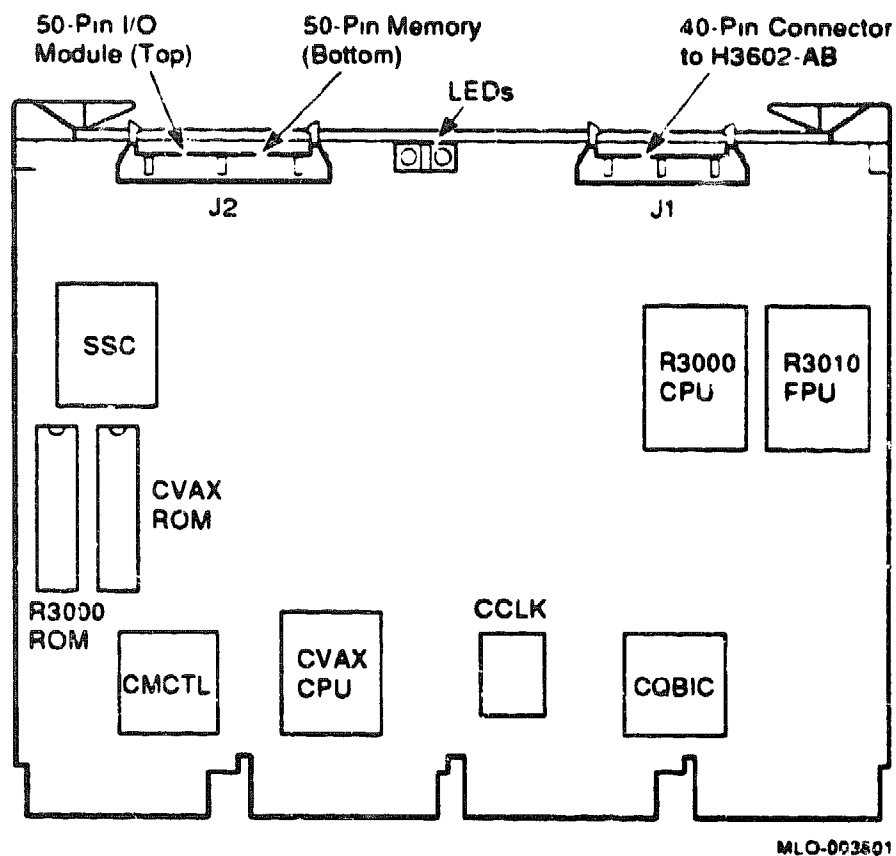
For a more detailed technical description of the KN210 CPU module, see the *KN210 CPU Module Set Technical Manual*.

## 1.1 KN210 Overview

The KN210 system supports only the ULTRIX-32 operating system (Version 3.1A and later) and is designed for applications that require high performance processing. KN210 configurations include the capability for server and multiuser support.

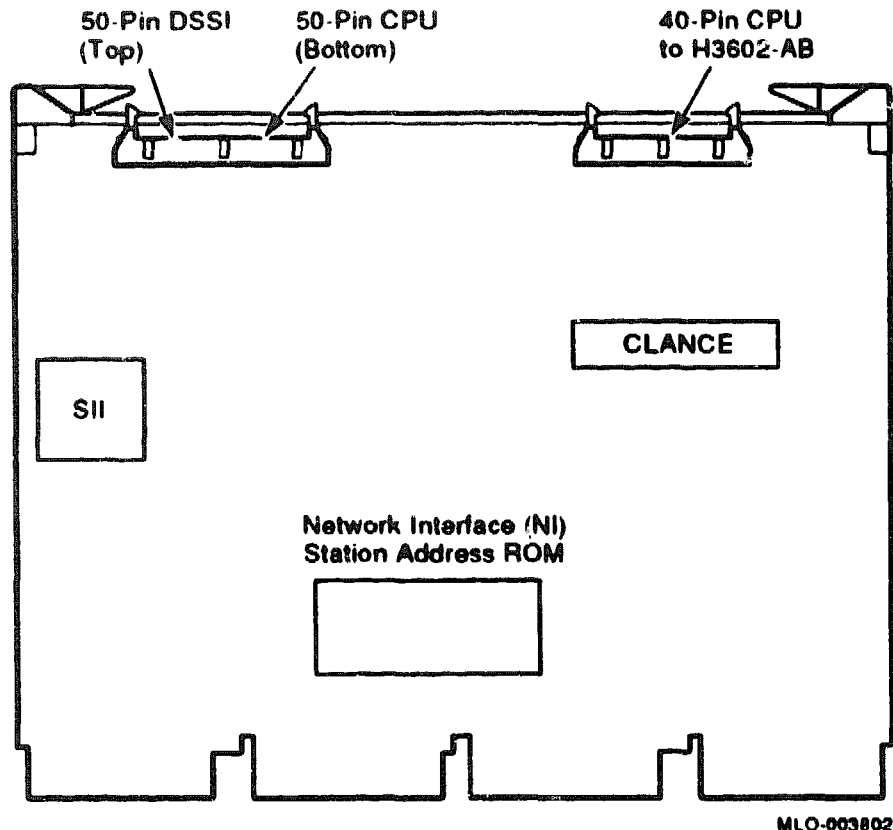
The KN210 CPU module (M7635-AA), shown in Figure 1-1, is a quad-height processor module. The KN210 CPU operates at 20 MHz and contains a reduced instruction set computer (RISC) processor based on the R3000 MIPSco chipset. The RISC implementation is based on a CPU architecture that uses a pipelined design, a simple instruction set, and write buffering.

**Figure 1-1: KN210 CPU Module (M7635-AA)**



The KN210 I/O module (M7636-AA), shown in Figure 1-2, is a quad-height module that contains mass storage and network interfaces. These interfaces provide higher performance than those available on the Q22-bus.

Figure 1-2: KN210 I/O Module (M7636-AA)

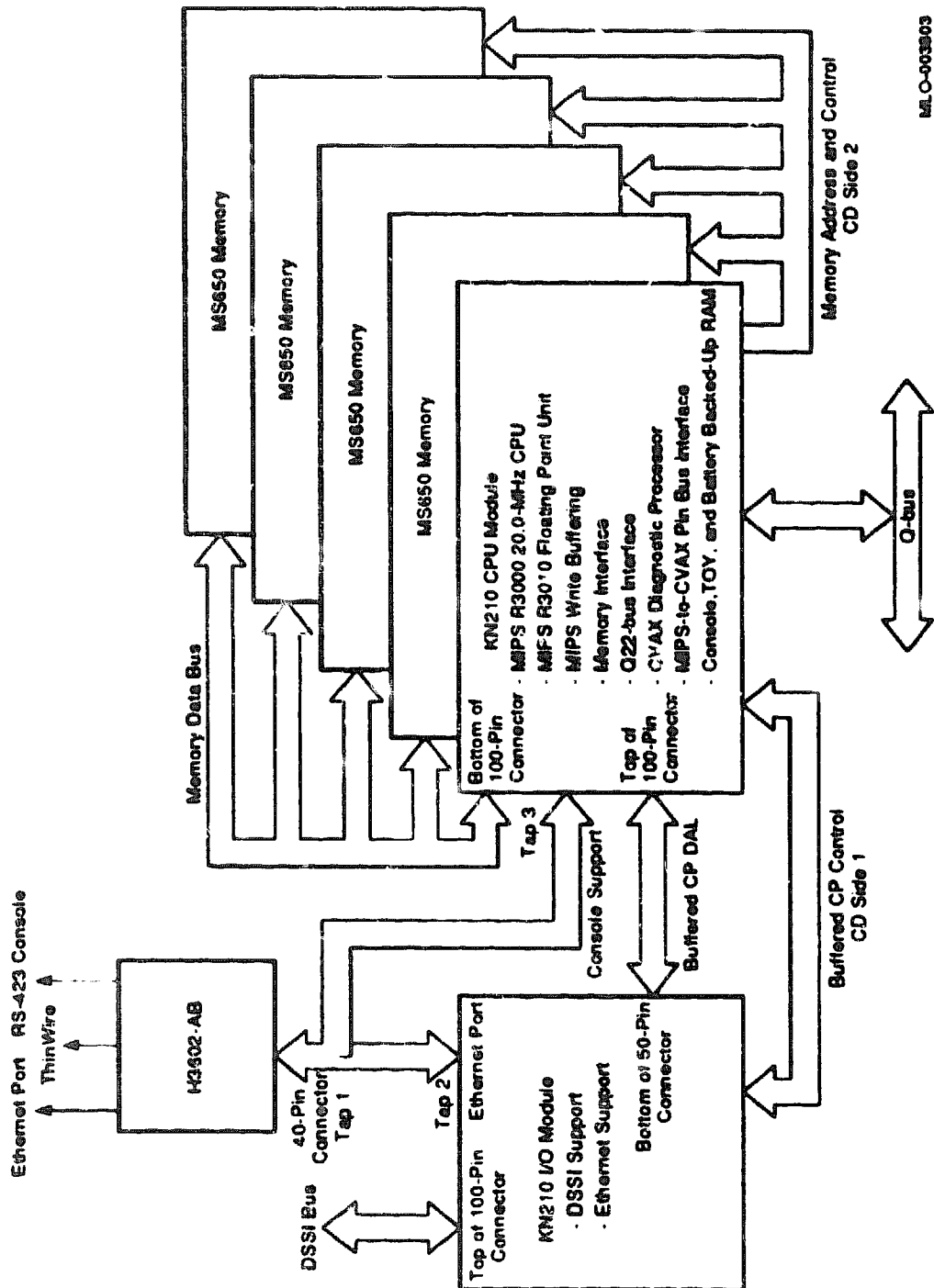


The KN210 I/O module appears as the following asynchronous slave devices on the KN210 CPU's buffered address and data lines:

- Ethernet station address ROM
- Ethernet controller chip (CLANCE)
- Ethernet buffer memory
- DSSI controller chip (SII)
- DSSI buffer memory

Figure 1-3 shows a functional block diagram of the KN210 CPU module set.

Figure 1-3: KN210 CPU Module Set Functional Block Diagram



The mass storage interface can control up to seven devices and consists of a Digital Storage System Interconnect (DSSI) port capable of a transfer rate of 4 Mbytes per second. The network interface is an Ethernet controller. Both interfaces have 128 Kbytes of static RAM for buffer space and use a 32-bit data and address path (CDAL) to the CPU module.

The KN210 CPU module and KN210 I/O module are used in two systems:

- DECsystem 5400 210QS (BA213 pedestal enclosure)
- DECsystem 5400 210QF (BA213 chassis in an H9644 cabinet)

Refer to *BA213 Enclosure Maintenance* and *H9644 Cabinet Maintenance* for a detailed description of each enclosure.

**CAUTION:** *Static electricity can damage integrated circuits. Use the wrist strap and antistatic mat found in the Antistatic Kit (29-26246) when you work with the internal parts of a computer system.*

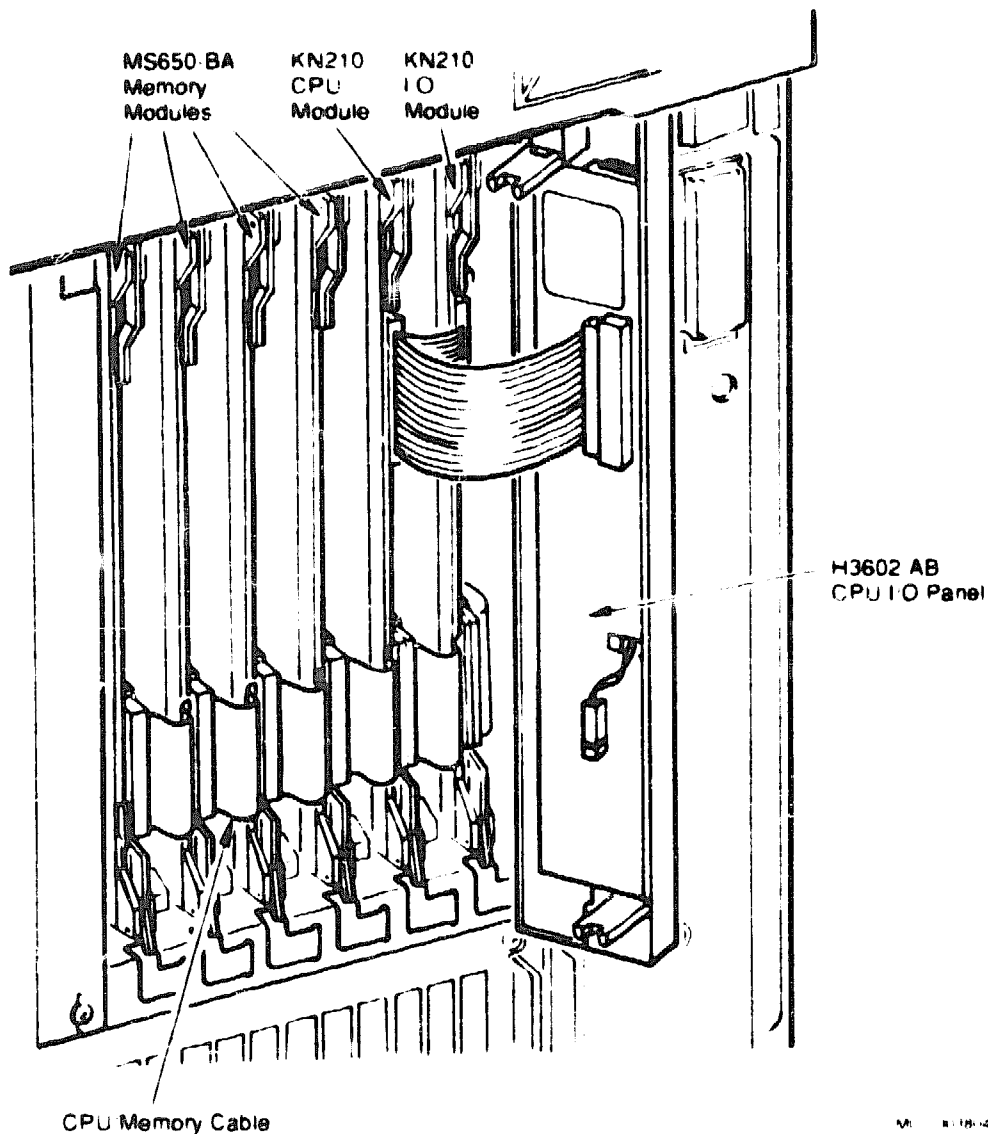
The KN210 CPU, the KN210 I/O module, and the MS650-BA memory module(s) combine to form a RISC CPU and memory subsystem that uses the Q22-bus to communicate with I/O devices. The KN210 and MS650-BA modules mount in standard Q22-bus backplane slots that implement the Q22-bus in the AB rows and the CD interconnect in the CD rows. The KN210 can support up to four MS650-BA modules (64 Mbytes maximum) if enough CD slots are available.

**DIFFERENCES:** *The KN210 system contains a two-module set: the KN210 CPU module and KN210 I/O module. The KN210 CPU module (M7635-AA) is always installed in slot 2 of the backplane, and the KN210 I/O module (M7636-AA) is always installed in slot 1.*

*Similar MicroVAX systems contain only the CPU module, which is always installed in slot 1 of the backplane. The I/O capabilities reside either on the CPU module or on a separate controller module.*

You install the KN210 I/O module in slot 1 of the BA213 backplane, the KN210 CPU module in slot 2, and MS650-BA memory modules in slots 3 through 6, as shown in Figure 1-4.

**Figure 1-4: Location of KN210 and Memory Modules In BA213 Card Cage**



1-6 1-6-4



## 1.2 KN210 Features

The major features of the KN210 CPU module set are as follows:

- A MIPS R3000 RISC processor with a cycle time of 50 ns
- A MIPS R3010 floating-point unit
- A 64-Kbyte, 25-ns instruction cache
- A 64-Kbyte, 25-ns data cache
- A DSSI mass storage interface
- An Ethernet interface
- A main memory controller that supports up to 64 Mbytes of error correction code (ECC) memory. The ECC memory resides on one-to-four MS650-BA memory modules, depending on the system configuration.
- A console port featuring the following:
  - Console terminal displays
  - Switch-selected baud rates
- A Q22-bus interface that supports up to 16-word, block mode transfers between a Q22-bus DMA device and main memory, and block mode transfers of up to 2 words between the CPU and Q22-bus devices. This Q22-bus interface contains:
  - A 16-entry map cache for the 8192 entry, scatter-gather map that resides in main memory. This map translates 22-bit, Q22-bus addresses into 26-bit main memory addresses.
  - Interrupt arbitration logic that recognizes Q22-bus interrupt requests BR7 through BR4.
  - 240-ohm, Q22-bus termination.
- A CVAX diagnostic processor with a cycle time of 100 ns.
- A 128-Kbyte ROM (R3000).
- A 128-Kbyte ROM (CVAX).

## 1.2.1 R3000 RISC Processor

The R3000 chip resides on the KN210 CPU module and implements two tightly coupled processors in a single VLSI chip. One processor is the 32-bit CPU and the other is the system control processor (CP0). The combined processors provide the following features:

- 32-bit operation. The R3000 contains thirty-two 32-bit registers that use 32-bit addressing.
- Pipelined design. The five-stage pipeline is capable of executing one instruction per 50-ns cycle.
- On-chip cache control. Separate instruction and data caches. Both caches can be accessed in a single CPU cycle.
- On-chip memory management. The 4-Gbyte virtual address space is mapped with a 64-entry, fully associative translation lookaside buffer (TLB).
- Coprocessor interface. A tightly coupled coprocessor interface for up to four coprocessors. CP0 is located on the CPU chip; CP1 is the floating-point accelerator. CP2 and CP3 are not used.
- Write buffers. Four R3020 four-word-deep write buffers. All writes pass through these write buffers.

**DIFFERENCES:** *For the KN210-based system, the terminology for various words is as follows:*

- *CVAX: a word consists of 16 bits, and a longword consists of 32 bits.*
- *R3000: a halfword consists of 16 bits, and a word consists of 32 bits. This terminology reflects the R3000 technology.*

*For similar MicroVAX systems, the terminology is the same as for the CVAX, as listed above.*

## 1.2.2 Floating-Point Accelerator

The KN210 floating-point accelerator resides on the KN210 CPU module and is implemented by a single VLSI chip called the R3010.

## 1.2.3 Cache Memory

To maximize CPU performance, the KN210 CPU module contains a 64-Kbyte instruction cache and a 64-Kbyte data cache. Both caches have the same organization and are direct mapped, with a block size of one word

(four bytes). The fill size is either one word (4 bytes) or four words (16 bytes).

### **1.2.4 Memory Controller**

The main memory controller resides on the KN210 CPU module and is implemented by a VLSI chip called the CMCTL, which supports ECC memory.

The maximum amount of main memory supported by KN210 systems is 64 Mbytes. This memory resides on from one-to-four MS650-BA memory modules, depending on the system configuration. The MS650 modules communicate with the KN210 through the MS650 memory interconnect, which uses the CD interconnect and a 50-pin, multiconnector ribbon cable.

### **1.2.5 Console Serial Line**

The console serial line provides the KN210 with a full-duplex, RS-423 EIA serial line interface, which is also RS-232C compatible. An 8-bit data format is supported, with no parity and one stop bit.

### **1.2.6 Time-of-Year Clock and Timers**

The KN210 CPU module clocks include the time-of-year clock (TODR), two additional programmable timers, and a 100-Hz interval timer that is used as the R3000 interval clock.

### **1.2.7 Boot and Diagnostic Facility**

The KN210 CPU boot and diagnostic facility consists of the following:

- Four registers
- Two 40-pin ROMs (128 Kbytes of read-only memory each)
- One Kbyte of battery backed-up RAM

The ROM and battery backed-up RAM may be accessed by longword, word, or byte references.

The resident firmware consists of 256 Kbytes of 16 bit-wide ROM, located on two EPROMs. The firmware gains control when the processor halts, and contains programs that provide the following services:

- Board initialization
- Power-up self-testing of the KN210 and MS650 modules
- R3000 console program

- Emulation of a subset of the VAX standard console (automatic or manual bootstrap, automatic or manual restart, and a simple command language for examining or altering the state of the processor)
- Booting from supported Q22-bus devices
- Multilingual capability

The firmware is described in detail in Chapter 3.

### 1.2.8 Q22-bus Interface

The KN210 CPU includes a Q22-bus interface, which is implemented by a single VLSI chip called the CQBIC. The CQBIC contains an interface between the CDAL bus and the Q22-bus and supports the following:

- A programmable mapping function (scatter-gather map) for translating 22-bit, Q22-bus addresses into 26-bit main memory addresses. This mapping function allows any page in the Q22-bus memory space to be mapped to any page in main memory.
- A direct mapping function for translating 26-bit main memory addresses into 22-bit, Q22-bus addresses. These main memory addresses are located in the local Q22-bus address space and the local Q22-bus I/O page.
- Masked and unmasked longword reads and writes from the CPU to the Q22-bus memory and I/O space and the Q22-bus interface registers.
  - Longword reads and writes of the local Q22-bus memory space are buffered and translated into two-word (16 bits), block mode transfers.
  - Longword reads and writes of the local Q22-bus I/O space are buffered and translated into two single-word transfers.
- Block-mode writes of up to 16 words from the Q22-bus to main memory.
- Transfers from the CPU to local Q22-bus memory space. The Q22-bus map translates the address back into main memory (local-miss, global-hit transactions).

### 1.2.9 CVAX Diagnostic Processor

The KN210 CPU diagnostic processor is implemented by a single VLSI chip called the CVAX. The CVAX supports the MicroVAX chip subset (plus six additional string instructions) of the VAX instruction set, data types, and full VAX memory management. The processor state is composed of 16 general purpose registers (GPRs), the processor status longword (PSL), and internal processor registers (IPRs).

### **1.2.10 Network Interface**

The KN210 I/O module contains a network interface implemented through a CLANCE chip, a 32K x 8 bit-wide ROM, and four 32K x 8-bit static RAMs. This interface allows you to connect the KN210 I/O module to either a ThinWire or standard Ethernet cable through the H3602-AB CPU I/O panel.

The network interface contains four registers for control and status reporting, a 24-word (48-byte) transmit silo, and a 24-word receive silo. It also contains the four 32K x 8 static RAMs. The DMA controller reads control information and writes status information to and from the buffer. The network interface transfers data (one word per memory reference) between buffer memory and either the transmit or receive silo. The DMA controller can perform up to eight masked longword references per burst. Each reference takes 600 ns and contains either a byte or word of data. The minimum time between bus requests is 8  $\mu$ sec.

### **1.2.11 Mass Storage Interface**

The KN210 I/O module contains an SII chip and four 32K x 8-bit static RAMs that implement the Digital Storage System Interconnect (DSSI) bus interface. The DSSI interface allows the KN210 to transmit packets of data to, and receive packets of data from, up to seven RF-series integrated storage elements (ISEs). The DSSI bus improves system performance for two reasons:

- It is faster than the Q22-bus.
- It relieves the Q22-bus of disk traffic, thereby allowing more bandwidth for Q22-bus devices.

The physical characteristics of the DSSI bus are as follows:

- 4 Mbytes/sec maximum bandwidth
- Distributed arbitration
- Synchronous operation
- Parity checking
- Six-meter total bus length (includes internal and external cabling)
- Single-ended bus transceivers
- Maximum of eight nodes (KN210 I/O module counts as one)
- Eight data lines

- One parity line
- Eight control lines

Refer to the following sections for more information about the DSSI bus and ISEs:

Section 2.5	Setting and changing DSSI node names, addresses, and unit numbers
Section 3.9.15	Console SET HOST command
Section 4.4	DSSI ISE acceptance testing
Section 4.8	RF-series resident diagnostics and local programs

## 1.3 H3602-AB CPU I/O Panel

The H3602-AB CPU I/O panel, shown in Figure 1-5, contains the following components:

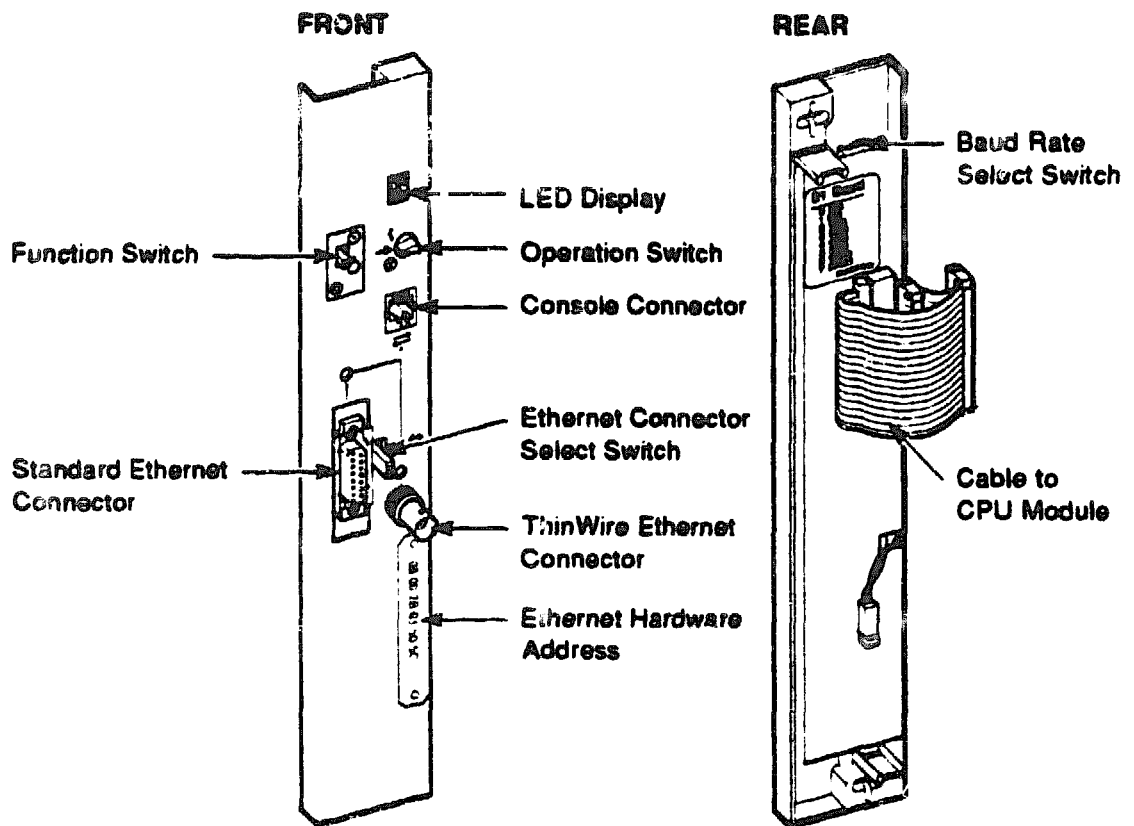
- An operation switch
- A function switch
- A console serial line connector
- A console baud rate select switch
- A 15-conductor connector for standard Ethernet cable
- A BNC plug connector and DESTA logic for a ThinWire Ethernet coaxial cable
- An Ethernet connector select switch
- One LED that indicates the selected Ethernet connector
- One LED that indicates valid +12 Vdc for the selected Ethernet connector

The H3602-AB switches are read by the firmware when the processor halts. For this reason, if you change the baud rate on the H3602-AB, the new baud rate does not take effect until you power up or reset the system or until you issue a break and continue.

The hex LED display shows the individual test numbers during the power-up self-tests and bootstrap. The LED display is described in Chapter 4, Table 4-7.

You connect the KN210 modules to the H3602-AB through a 40-pin, multi-connector ribbon cable (Figure 1-5).

**Figure 1-5: H3602-AB CPU I/O Panel**



MLO-002737

**DIFFERENCES:** *The meanings of the H3602-AB operation switch positions and function switch positions are different than for similar MicroVAX systems.*

*For the KN210 system, use the switch settings that appear in Table 1-1.*

The operation switch (three-position rotary) and the function switch (two-position slider) are described in Table 1-1. See Chapter 3 for a detailed description of power-up procedures and console commands.

**Table 1-1: H3602-AB Operation and Function Switch Settings**

Operation Switch Position	Function Switch Position	Action
{ ○ Action mode	○	Test. The console serial line external loopback test is executed at the completion of the power-up self-tests. Use an H3103 (12-25083-01) loopback connector.
	○	Query. The user is prompted for the console language at the completion of the power-up self-tests.
- Normal mode		Power-up self-tests run and console enters normal mode (>>).
		If the <i>bootmode</i> environment variable is set to <i>a</i> , the R3000 processor attempts to locate a booting device specified through the bootpath environmental variable.
		If the <i>bootmode</i> environment variable is not initialized or is set to <i>d</i> , the R3000 processor enters normal mode and prompts the user for commands.
	○	Breaks enabled.
⊕ Maintenance mode	○	Breaks disabled.
		Power-up self-tests run and console enters maintenance mode (>>>).
	○	Breaks enabled.
	○	Breaks disabled.

**DIFFERENCES:** *The KN210 system has no autoboot capability when the operation switch is set to the maintenance position. See Chapter 3, Section 3.5.1, for information on the ULTRIX-32 bootstrap procedure.*

*Similar MicroVAX systems do have the autoboot capability when the operation switch is set to the maintenance position.*



## **1.4 MS650-BA Memory**

The MS650-BA (M7622-AA) is a 16-Mbyte memory module that provides memory for the KN210. The MS650-BA is a nonintelligent memory array module controlled by a custom memory controller chip (CMCTL) on the KN210 CPU module.

The quad-height MS650-BA, shown in Figure 1-6, has a 450-ns, 39 bit-wide array (32-bit data and 7-bit ECC), implemented with 1-Mbit dynamic RAMs in dual in-line packages (DIPs).

The KN210 CPU module and up to four MS650-BA memory modules (64 Mbytes maximum) communicate through the MS650 memory interconnect. This interconnect uses the CD backplane interconnect for address and control signals and a 50-pin ribbon cable for data signals.

### **Ordering Information**

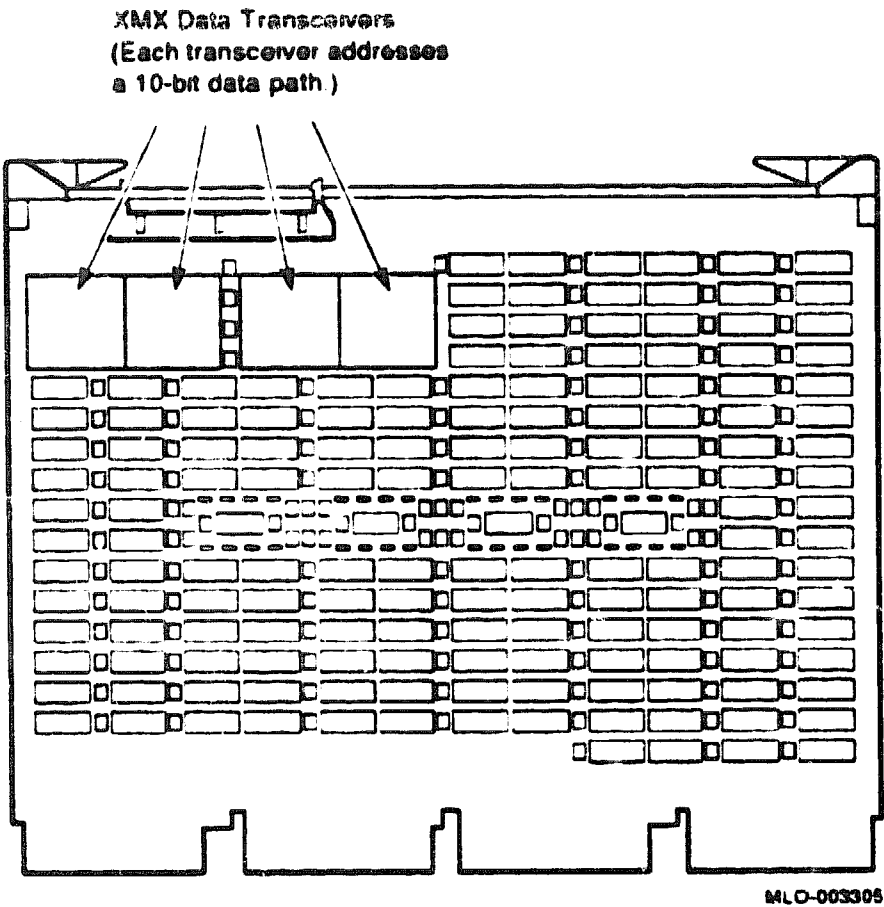
MS650-BA	16-Mbyte module only (M7622-AA).
MS650-BF	Option installation kit for BA200-series enclosures. Includes MS650-BA, filler panel assemblies, blank cover, CPU cable, labels, and installation guide.

### **Diagnostic Support**

MicroVAX Diagnostic Monitor  
Self-test

Release 128 (Version 3.11)  
KN210 self-test

**Figure 1-6: MS650-BA Memory Module (M7622-AA)**



MLO-003305

[illegible]

# CHAPTER 2

[illegible]

## Chapter 2

# Configuration

---

### 2.1 Introduction

This chapter describes the guidelines for changing the configuration of a KN210 system.

Before you change the system configuration, you must consider the following factors:

- Module order in the backplane
- Module configuration
- Mass storage device configuration

If you are adding a device to a system, you must know the capacity of the system enclosure in the following areas:

- Backplane
- I/O panel
- Power supply
- Mass storage devices

### 2.2 General Module Order

The order of modules in the backplane depends on four factors:

- Relative use of devices in the system
- Expected performance of each device relative to other devices
- The ability of a device to tolerate delays between bus requests and bus grants (called delay tolerance or interrupt latency)
- The tendency of a device to prevent other devices farther from the CPU from accessing the bus

## 2.3 Module Order for KN210 Systems

Observe the following rules about module order:

- Install the KN210 I/O module in slot 1.
- Install the KN210 CPU module in slot 2.
- Install an MS650-BA memory module in slot 3. Install any additional MS650-BA memory modules in slots 4, 5, and 6.
- Do not install dual-height modules in the CD rows.

The Q22-bus does not pass through the CD rows of the backplane in a BA200-series enclosure. Install all dual-height Q22-bus modules in the AB rows. Install dual-height grant cards only in the AB rows, or single-height grant cards only in the A row.

## 2.4 Module Configuration

Each module in a system must use a unique device address and interrupt vector. The device address is also known as the control and status register (CSR) address. Most modules have switches or jumpers for setting the CSR address; most interrupt vector values are set by software. The value of a floating address depends on what other modules are housed in the system.

Set CSR addresses and interrupt vectors for a module as follows:

1. Determine the correct values for the module with the **CONFIGURE** command at the maintenance mode prompt (**>>>**). Type **CONFIGURE**, then **HELP** for the list of supported devices.

**NOTE:** *Some of the devices listed in the HELP display are not supported by the KN210 CPU module set.*

**>>> CONFIGURE**

Enter device configuration, **HELP**, or **EXIT**

Device, Number? **help**

Devices:

LPV11	RXJ11	DLV11J	DZQ11	DZV11	DFA01
RLV12	TSV05	RXV21	DRV11W	DRV11B	DPV11
DMV11	DELQA	DEQNA	DESQA	RQDX3	KDA50
RRD50	RQC25	KFQSA-DISK	TQK50	TQK70	TU81E
RV20	KFQSA-TAPE	KMV11	IEQ11	DHQ11	DHV11
CXA16	CXB16	CXY08	VCB01	QVSS	LNV11
LNV21	QPSS	DSV11	ADV11C	AAV11C	AXV11C
KWV11C	ALV11D	AAV11D	VCB02	QDSS	DRV11J
DRQ3B	VSV21	IBQ01	IDV11A	IDV11B	IDV11C
IDV11D	IAV11A	IAV11B	MIRA	ADQ32	DTC04
DESNA	IGQ11				

Numbers:

1 to 255, default is 1

Device, Number?

See the description of the **CONFIGURE** command in Section 3.9.2 for an example of how to use this command to obtain the correct CSR addresses and interrupt vectors

The **LPV11-SA**, which is the **LPV11** version compatible with the **BA200**-series enclosures, has two sets of CSR address and interrupt vectors. To determine the correct values for an **LPV11-SA**, enter **LPV11, 2** at the **Device, Number?** prompt for one **LPV11-SA**, or enter **LPV11, 4** for two **LPV11-SA** modules

2. See Appendix C for instructions on how to configure the **KFQSA** storage adapter. Appendix C explains how to do the following:
  - Set a four-position switchpack on the **KFQSA**
  - Program the CSR addresses for all the system's **DSSI** devices into the **EEROM** on the **KFQSA**
  - Reprogram the **EEROM** when you add **DSSI** devices
3. See *Microsystems Options* for switch and CSR and interrupt vector jumper settings for supported options.

## 2.5 DSSI Configuration

Each device must have a unique Digital Storage System Interconnect (DSSI) node ID. The RF71 integrated storage element (ISE) receives its node ID from a plug on the operator control panel (OCP) on the front panel. By convention, ISEs are mounted in the BA213 enclosure from right to left, as listed in Table 2-1.

**Table 2-1: ISE Order In BA213 Enclosure**

Device	Position	Node ID <sup>1</sup>
First	Right side	0
Second	Center	1
Third	Left side	2

---

<sup>1</sup>KN210 node ID = 7

If the cable between the RF71 and the OCP is disconnected, the RF71 reads the node ID from three DIP switches on its electronics controller module (ECM).

**NOTE:** Pressing the system reset button on the front of a BA213 power supply has no effect on the RF71 ISEs. You must perform a power cycle.

The node ID switches are located behind the 50-pin connector on the ECM. Switch 1 (the MSB) is nearest to the connector. Switch 3 (the LSB) is farthest from the connector. Refer to the RF71 section in *Microsystems Options* for an illustration and further information. Table 2-2 lists the switch settings for the eight possible node addresses.

**Table 2-2: RF71 DIP Switch Settings**

Node ID	S1	S2	S3
0	Down	Down	Down
1	Down	Down	Up
2	Down	Up	Down
3	Down	Up	Up
4	Up	Down	Down
5	Up	Down	Up
6	Up	Up	Down
7	Up	Up	Up

The maintenance mode SET HOST/DUP command creates ISE device names according to the following scheme:

nodename \$ DIA unit number. For example, SUSAN\$DIA3

You can use the device name for booting, as follows:

```
>>> BOOT SUSAN$DIA3
```

You can access local programs in the RF71 through the MicroVAX Diagnostic Monitor (MDM) or through the maintenance mode SET HOST/DUP command. This command creates a virtual connection to the storage device and the designated local program, using the Diagnostic and Utilities Protocol (DUP) standard dialog. Section 3.9.15 describes the SET HOST/DUP command.

### 2.5.1 Changing the Node Name

Each RF71 ISE has a node name that is maintained in the EEPROM on the controller module. This node name is determined in manufacturing from an algorithm based on the drive serial number. You can change the node name of the ISE to something more meaningful by following the procedure in Example 2-1. In the example, the node name for the RF71 ISE at DSSI node address 1 is changed from R3YBNE to DATADISK.

See Section 4.8.5 for further information about the PARAMS local program.

#### Example 2-1: Changing a DSSI Node Name

```
>>> SHO DSSI
DSSI Node 0 (MDC)
-DIA0 -rf(0,0,*) (RF71)

DSSI Node 1 (R3YBNE)      !The node name for this drive will be
-DIA1 -rf(1,1,*) (RF71)  !changed from R3YBNE to DATADISK.
```

```
DSSI Node 7 (*)
```

```
>>>
```

```
>>> SET HOST/DUP/DSSI 1 PARAMS
```

```
Starting DUP server...
```

```
Copyright 1988 Digital Equipment Corporation
```

```
PARAMS> SHO NODENAME
```

Parameter	Current	Default	Type	Radix
NODENAME	R3YBNE	RF71	String	Ascii B

Example 2-1 Cont'd on next page



## Example 2-1 (Cont.): Changing a DSSI Node Name

```
PARAMS> SET NODENAME DATADISK

PARAMS> WRITE                                'This command writes the change
                                              'to EEPROM.

Changes require controller initialization, ok? [Y/(N)] y

Stopping DUP server...
>>> SHO DSSI
DSSI Node 0 (MDC)
-DIA0 -rf(0,0,*) (RF71)

DSSI Node 1 (DATADISK)                        'The node name has changed from
-DIA1 -rf(1,1,*) (RF71)                      'R3YBNE to DATADISK.

DSSI Node 7 (*)
```

### 2.5.2 Changing the Unit Number

By default, the RF71 assigns the ISE's unit number to the same value as the DSSI node address for that drive. This occurs whether the DSSI node address is determined from the OCP bus ID plugs or from the three DIP switches on the RF71 controller module.

RF71 ISEs conform to the Digital Storage Architecture (DSA). Each drive can be assigned a unit number from 0 to 16,383 (decimal). The unit number need not be the same as the DSSI node address.

Example 2-2 shows how to change the unit number of an ISE. This example changes the unit number for the RF71 at DSSI node address 1 from 1 to 14 (decimal). You must change two parameters: UNITNUM and FORCEUNI. Changing these parameters overrides the default, which assigns the unit number the same value as the node address.

See Section 4.8.5 for further information about the PARAMS local program.

## Example 2-2: Changing a DSSI Unit Number

```
>>> SHO DSSI
DSSI Node 0 (MDC)
-DIA0 -rf(0,0,*) (RF71)

DSSI Node 1 (R3QJNE)      !The unit number for this drive will be
-DIA1 -rf(1,1,*) (RF71)  !changed from 1 to 14 (DIA1 to DIA14).

DSSI Node 7 (*)
>>>
>>> SET HOST/DUP/DSSI 1
Starting DUP server...
Copyright 1988 Digital Equipment Corporation
DRVEXR V1.0 D 2-JUN-1989 15:33:06
DRVST V1.0 D 2-JUN-1989 15:33:06
HISTRY V1.0 D 2-JUN-1989 15:33:06
ERASE V1.0 D 2-JUN-1989 15:33:06
PARAMS V1.0 D 2-JUN-1989 15:33:06
DIRECT V1.0 D 2-JUN-1989 15:33:06
End of directory

Task Name? PARAMS
Copyright 1988 Digital Equipment Corporation

PARAMS> SHO UNITNUM

Parameter      Current      Default      Type      Radix
-----
UNITNUM        1            1            Word      Dec      U

PARAMS> SHO FORCEUNI

Parameter      Current      Default      Type      Radix
-----
FORCEUNI        1            1            Boolean   0/1      U

PARAMS> SET UNITNUM 14
PARAMS> SET FORCEUNI 0
PARAMS> WRITE      !This command writes the changes
                   !to the EEPROM.

PARAMS> EX
Exiting...

Task Name?
```

Example 2-2 Cont'd on next page

## Example 2-2 (Cont.): Changing a DSSI Unit Number

Stopping DUP server...

>>>

>>> SHOW DSSI

DSSI Node 0 (MDC)

-DIA0 -rf(0,0,\*) (RF71)

DSSI Node 1 (R3QJNE)

!The unit number has changed

-DIA14 -rf(1,14,\*) (RF71)

!and the node ID remains at 1.

DSSI Node 7 (\*)

### 2.5.3 DSSI Cabling

A 50-conductor, multiconnector ribbon cable connects the RF71 ISE to the DSSI bus (Figure 2-1). A separate 5-conductor cable carries +5 Vdc and +12 Vdc to the ISE from the enclosure power supply.

A 10-conductor cable connects the ISE to the operator control panel (OCP, Figure 2-2). In the BA213 enclosure, there are two cables that connect the power supplies to the ISEs:

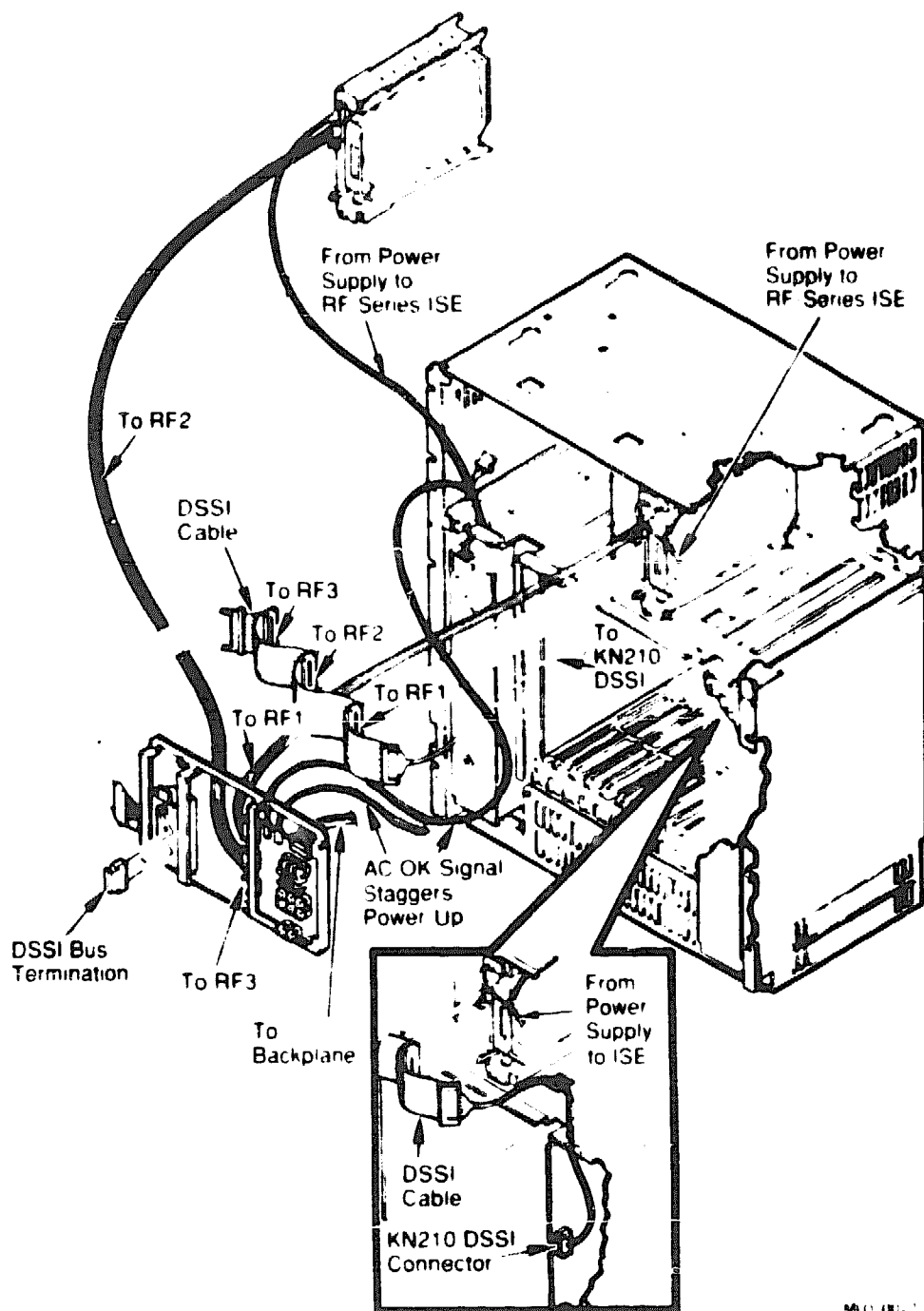
- One cable connects the ISEs to the power supply located on the right side, which supplies power to the two right-side mass storage devices.
- One cable connects the ISEs to the power supply located on the left side, which supplies power to the two left-side mass storage devices.

These cables carry the ACOK signal (same as POK) to each ISE. The OCP delays this signal to one ISE for each power supply to stagger the start-up of one of two possible devices attached to each supply. This delay prevents the ISEs from drawing excessive current from the power supplies at power-up.

The 50-conductor, multiconnector DSSI ribbon cable connects to a 50-conductor round cable that is routed through the bottom of the mass storage area to the DSSI connector on the KN210 I/O module.

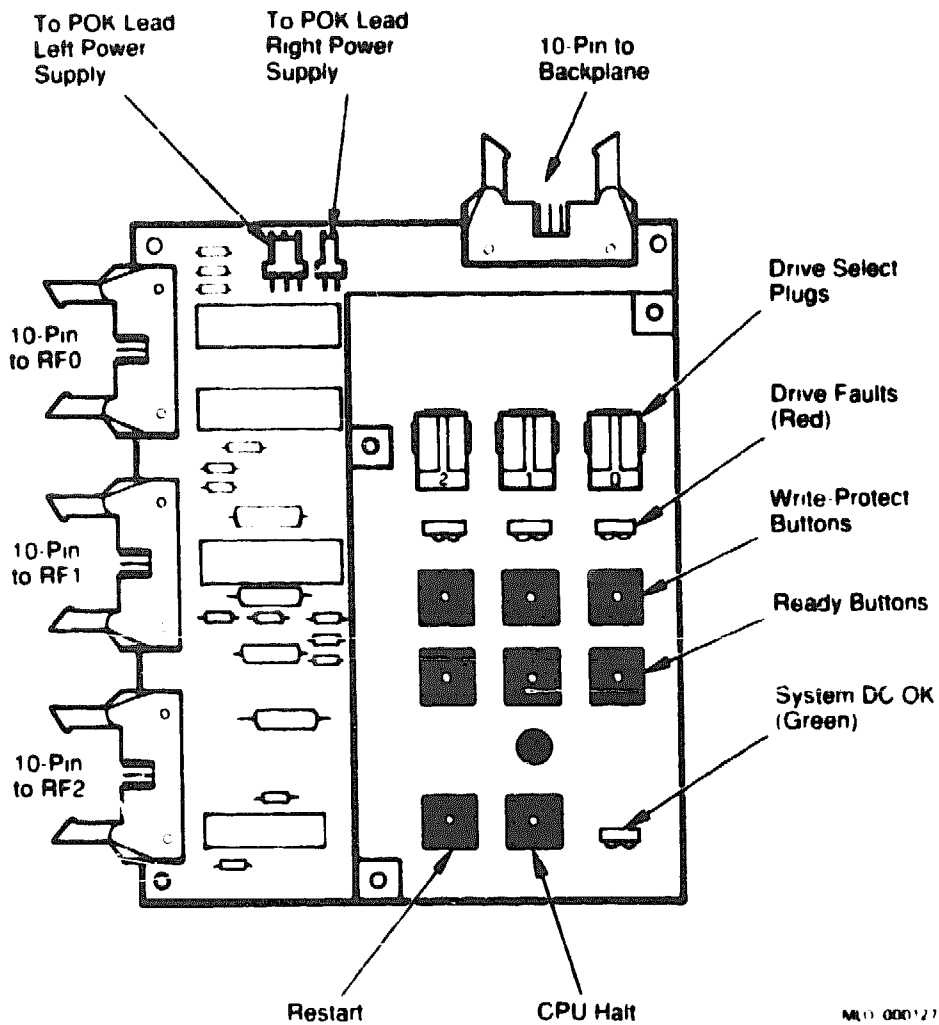
**CAUTION:** When removing or installing new ISEs, be sure to connect the rightmost connector of the DSSI multiconnector ribbon cable to the round cable coming from the KN210 I/O module. Do not "T" the bus by connecting the round connector to any of the ribbon cable's center connectors.

**Figure 2-1: DSSI Cabling, BA213 Enclosure**



WAL-001, 114

**Figure 2-2: HF-Series OCP**



The DSSI bus has a maximum length of 6 m (19.8 ft), including internal and external cabling. The DSSI bus must be terminated at both ends. The KN210 I/O module terminates the DSSI bus at one end. A 50-pin Honda terminator on the left side of the media faceplate terminates the bus at the other end. This terminator can be removed if you need to expand the bus.

## 2.6 Configuration Worksheet

This section provides a configuration worksheet for the BA213 system enclosure (Figure 2-3). Use the worksheet to make sure the configuration does not exceed the system's limits for expansion space, I/O space, and power.

Table 2-3 lists power values for supported devices. To check a system configuration, follow these steps:

1. List all the devices already installed in the system.
2. List all the devices you plan to install in the system.
3. Fill in the information for each device, using the data listed in Table 2-3.
4. Add up the columns. Make sure the totals are within the limits for the enclosure.
5. Using the procedure described in Section 2.4, confirm CSR address and vectors for all modules installed in the backplane.

In the BA213 enclosure, you must install a quad-height load module (M9060-YA) in one of backplane slots 7 through 12 if the continuous minimum current drawn on the second power supply is less than 5 amperes. If the minimum current of 5 amperes is not reached, the power supply enters an error mode and shuts down the system.

**Table 2-3: Power and Bus Loads for KN210 Options**

Option	Module	Current (Amps)		Power	Bus Loads	
		+5 V	+12 V	Watts	AC	DC
CXA16-AA-AF	M3118-YA	1.6	0.20	10.4	3.0	0.5
CXB16-AA-AF	M3118-YB	2.0	0.0	10.0	3.0	0.5
CXY08-AA-AF	M3119-YA	1.64	0.395	12.94	3.0	0.5
DELQA-SA	M7516-FA	2.7	0.5	19.5	3.3	0.5
DESQA-SA	M3127-PA	2.4	0.2	14.6	3.3	0.5
DSV11-SA	M3108-PA	5.43	0.69	35.4	3.9	1.0
KDA50-Q	M7164	6.93	0.0	34.6	3.0	0.5
KDA50-Q	M7165	6.57	0.03	33.21	-	-
KFQSA-SA	M7769	5.5	0.0	27.5	4.4	0.5
KLESI-SA	M7740-PA	4.0	0.0	20.0	5.0	1.0
KN210-AA	M7635-AA	4.5	0.13	24.0	3.5	1.0
KN210-AA	M7636-AA	5.0	0.23	27.8	0.0	0.0
LPV11-SA	M8080-PA	2.8	0.0	14.0	1.8	0.5
M9060-YA	-	5.3	0.0	26.5	0.0	0.0
MS650-BA	M7622-AA	1.1	0.0	5.5	0.0	0.0
RF30	-	1.10	0.80	15.1	-	-
RF71E-SA	-	1.25	4.54	26.5	-	-
TK70E-EA	-	1.5	2.4	36.3	-	-
TQK70-SA	M7559	3.5	0.0	17.5	4.3	0.5

**Figure 2-3: BA213 Configuration Worksheet**

**RIGHT POWER SUPPLY**

SLOT	MODULE	Current (Amps)		Power (Watts)
		+5 Vdc	+12 Vdc	
1				
2				
3				
4				
5				
6				
<b>MASS STORAGE:</b>				
	TK Drive			
	FIXED DISK 0			
	Total these columns:			
	Must not exceed:	33.0 A	7.6 A	230.0 W

**LEFT POWER SUPPLY**

SLOT	MODULE	Current (Amps)		Power (Watts)
		+5 Vdc	+12 Vdc	
7				
8				
9				
10				
11				
12				
<b>MASS STORAGE:</b>				
	FIXED DISK 1			
	FIXED DISK 2			
	Total these columns:			
	Must not exceed:	33.0 A	7.6 A	230.0 W

MLO-003805



[illegible]

# CHAPTER 3

[illegible]

## Chapter 3

# KN210 Firmware

---

### 3.1 Introduction

This chapter describes the KN210 firmware.

The firmware is located in two 128-Kbyte EPROMS (one per processor) on the KN210. The EPROMs are arranged as 32-bit words and are located at the following R3000 and CVAX restart locations:

- R3000: physical address 1FC00000
- CVAX: physical address 20040000

**DIFFERENCES:** *In the KN210 system, the firmware resides in two EPROMS: R3000 and CVAX. In similar MicroVAX systems, the firmware resides in only the CVAX EPROM.*

The CVAX and R3000 firmware contain the major functional blocks of code listed in Table 3-1.

**Table 3-1: KN210 Firmware Code**

CVAX Firmware Code	R3000 Firmware Code
Halt entry, dispatch, and exit	Primary and secondary bootstrap (ULTRIX-32)
Primary and secondary bootstrap (MDM)	Console program (normal mode commands)
Console program (maintenance mode commands)	
System restart	
ROM-based diagnostics	

**This chapter discusses the following**

- **CVAX halt procedures**
- **Power-up procedures**
- **Bootstrap procedures for ULTRIX-32 and MDM**
- **R3000 console program and normal mode commands**
- **CVAX console program and maintenance mode commands**

**The CVAX ROM-based diagnostics are described in Chapter 4.**

## **3.2 KN210 Firmware Features**

**The KN210 CVAX and R3000 firmware provide the following features:**

- **Automatic or manual bootstrap of customer application images at power-up, reset, or conditionally after processor halts.**
- **A CVAX interactive command language (maintenance mode) that allows you to examine and alter the state of the processor.**
- **An R3000 interactive command language (normal mode) that allows you to make use of environment variables to pass information to the ULTRIX-32 operating system.**
- **Diagnostics and console utilities that test all components on the KN210 and memory modules and perform extended tests on the DSSI bus and Ethernet. The KN210 firmware also provides RF-series ISEs and the H3602-AB CPU I/O panel, to verify that the components are working correctly.**
- **LEDs and displays on the KN210 CPU module and console terminal that display diagnostic progress and error reports.**
- **Multilingual support. The firmware can issue system messages in several languages.**

**The processor must be functioning at a level capable of executing instructions from the maintenance program ROM for the maintenance program to operate.**

### 3.3 CVAX Halt Entry and Dispatch Code

The CVAX processor enters the halt entry code at physical address 20040000 whenever a halt occurs. The halt entry code saves machine state, then transfers control to the firmware halt dispatcher.

After a halt, the halt entry code saves the current LED code, then writes an E to the LEDs. An E on the LEDs indicates that at least several instructions have been successfully executed, although if the CPU is functioning properly, it occurs too quickly to be seen. The halt entry code saves the following registers. The console intercepts any direct reference to these registers and redirects it to the saved copies:

R0-R16	General purpose registers
PR\$_SAVPSL	Saved processor status longword register
PR\$_SCBB	System control block base register
DLEDR	Diagnostic LED register
SSCCR	SSC configuration register
ADxMCH	SSC address match register
ADxMSK	SSC address mask register

The halt entry code unconditionally sets the following registers to fixed values on any halt, to ensure that the console itself can run.

SSCCR	SSC configuration register
ADxMCH	SSC address match register
ADxMSK	SSC address mask register
CBTCR	CDAL bus timeout control register
TIVRx	SSC timer interrupt vector registers

The console command interpreter does not modify actual processor registers. Instead it saves the processor registers in console memory when it enters the halt entry code, then directs all references to the processor registers to the corresponding saved values, not to the registers themselves.

When the processor reenters normal mode, the saved registers are restored and any changes become operative only then. References to processor memory are handled normally. The binary load and unload command (X, Section 3.9.20) cannot reference the console memory pages.

After saving the registers, the halt entry code transfers control to the halt dispatch code. The halt dispatch code determines the cause of the halt by reading the halt field (PR\$\_SAVPSL <13:08>), the processor halt action field (PR\$\_CPMBX <01:00>), and the break enable switch on the H3602-AB panel. Table 3-2 lists the actions taken, by sequence. If an action fails, the next action is taken, with the exception of bootstrap, which is not attempted after diagnostic failure.

**Table 3-2: Actions Taken on a Halt**

<b>Breaks Enabled on H3602-AB</b>	<b>Power-Up Halt<sup>1</sup></b>	<b>Halt Action<sup>2</sup></b>	<b>Action</b>
T <sup>3</sup>	T	X	Diagnostics, halt
T	F	0	Halt
F	T	X	Diagnostics, halt
F	F	0	Restart, halt
X	F	1	Restart, halt
X	F	2	Halt
X	F	3	Halt

<sup>1</sup>Power-up halt: PR\$ \_SAVPSL<13:00>=3

<sup>2</sup>Halt action: PR\$ \_CPMBX<01:00>

<sup>3</sup>T = condition is true, F = condition is false, X = does not matter

Several conditions can trigger an external halt, and different actions are taken depending on the condition. The conditions are listed below.

- The function switch is set to enable, and you press **Break** on the system console terminal.
- Assertion of the BHALT line on the Q22-bus
- Deassertion of DCOK. A halt is delivered if the processor is not running out of halt-protected space, and the BHALT ENB bit is set. The system restart switch deasserts DCOK. DCOK may also be deasserted by the DELQA sanity timer, or any other Q22-bus module that chooses to implement the Q22-bus restart/reboot protocol.

When in maintenance mode, the processor halts on the deassertion of DCOK. If halts are enabled, the firmware enters maintenance mode. If halts are disabled, the firmware takes the action dictated by the halt action field.

The action taken by the halt dispatch code on a console **Break** or Q22-bus BHALT is the same: the firmware enters maintenance mode if halts are enabled.

The halt dispatch code distinguishes between DCOK deasserted and BHALT by assuming that BHALT must be asserted for at least 10 msec, and that DCOK is deasserted for at most 9 µsec. To determine if the BHALT line is asserted, the firmware steps out into halt-unprotected space after 9 msec. If the processor halts again, the firmware concludes that the halt was caused by the BHALT and not by the deassertion of DCOK.

## 3.4 Power-Up

On power-up, the CVAX firmware performs several unique actions. It runs the initial power-up test (IPT), locates and identifies the console device, performs a language inquiry, and runs the remaining diagnostics.

The IPT waits for power to stabilize by monitoring SCR<5>(POK). Once power is stable, the IPT verifies that the console private nonvolatile RAM (NVRAM) is valid (backup battery is charged). If it is invalid or zero (battery is discharged), the IPT tests and initializes the NVRAM.

After the battery check, the firmware tries to determine the type of terminal attached to the console serial line. If the terminal is a known type, it is treated as the system console.

Once a console device has been identified, the firmware displays the KN210 banner message:

```
KN210-A Vn.n
```

The banner message contains the processor name (KN210-A) and the version of the firmware (Vn.n), where n.n denotes the major and minor release numbers.

Power-up actions differ, depending on the state of the operation switch located on the H3602-AB CPU I/O panel, shown previously in Figure 1-5.

### 3.4.1 Power-Up Sequence: Operation Switch Set to Normal

If you set the operation switch on the H3602-AB CPU I/O panel to the normal position (→), the power sequence is as follows:

1. CVAX powers up (begins execution at a location pointed to by physical address 20040000).
2. CVAX runs self-test diagnostics.
3. CVAX executes an EXIT command (a 1 is written into the SPR).
4. CVAX hangs on a DMA grant.
5. R3000 begins execution at an address stored in 1FC00000.
  - If the *bootmode* environment variable is set to *a*, the R3000 attempts to autoboot. If the *bootpath* environment variable is valid, the autoboot succeeds. See Section 3.6.2 for a description of environment variables.

- If the *bootmode* environment variable is not initialized (\*) or if it is set to *d*, the R3000 prompts you for a command at the normal mode prompt (>>)

If you enter *maint* at the >> prompt (a zero is written in the SPR), the R3000 hangs on a RDBUSY stall and the CVAX resumes execution (the >>> prompt is displayed).

In addition, the console displays the language selection menu if the operation switch is set to the normal position (•) and the contents of NVRAM are invalid. The console uses the saved console language if the operation switch is set to the normal position and the contents of NVRAM are valid.

### 3.4.2 Power-Up Sequence: Operation Switch Set to Maintenance

If you set the operation switch on the H3602-AB CPU I/O panel to the maintenance position (⊕), the power-up sequence is as follows:

1. CVAX powers up (begins execution at a location pointed to by physical address 20040000).
2. CVAX runs self-test diagnostics
3. CVAX enters maintenance mode and prompts you for commands at the maintenance mode prompt (>>>).
  - If you enter *EXIT* at the >>> prompt, the CVAX hangs on a DMA grant and the R3000 begins execution (the >> prompt is displayed).

### 3.4.3 Operation Switch Set to Action: Loopback Tests

You can verify the connection between the KN210 CPU module set and the console terminal, as follows:

1. Set the operation switch to the action position (⊖).
2. Set the function switch to enable (••).
3. To test the console port, connect the H3103 loopback connector to the H3602-AB console connector. (You must install the loopback connector for the test to run.)
4. To test the console cable, connect the H8572 connector to the end of the console cable, and connect the H3103 to the H8572

During the test, the firmware toggles between the active and passive states. During the active state (3 seconds), the LED is set to 7. The firmware reads the baud rate and operation switch setting, then transmits and receives a character sequence.

During the passive state (7 seconds), the LED is set to 4.

If at any time the firmware detects an error (parity, framing, overflow, or no characters), the display hangs at 7. If you move the operation switch from the action position, the firmware continues as on a normal power-up.

### **3.4.4 Operation Switch Set to Action: Language Query**

If you set the operation switch to the action position (ⓘ) and the function switch to query (Ⓚ), or if the firmware detects that the contents of NVRAM are invalid, the firmware prompts you for the language to be used for displaying the following system messages:

- Loading system software.
- Failure.
- Restarting system software.
- Performing normal system tests.
- Tests completed.
- Normal operation not possible.
- Bootfile.

The selection menu for the language and keyboard type is shown in Example 3-1. If no response is received within 30 seconds, the firmware defaults to English.

**NOTE:** *Some older terminals, such as the VT100, do not support multiple languages. In these terminals, the language selection menu does not appear and the system defaults to English.*



## Example 3-1: Language Selection Menu

KN210-A X0.2-9

- 1) Dansk
  - 2) Deutsch (Deutschland/Österreich)
  - 3) Deutsch (Schweiz)
  - 4) English (United Kingdom)
  - 5) English (United States/Canada)
  - 6) Español
  - 7) Français (Canada)
  - 8) Français (France/Belgique)
  - 9) Français (Suisse)
  - 10) Italiano
  - 11) Nederlands
  - 12) Norsk
  - 13) Português
  - 14) Suomi
  - 15) Svenska
- (1..15):

## 3.5 Bootstrap

Bootstrapping is the process of loading and transferring control to an operating system. The KN210 bootstrap support is as follows:

- **CVAX** (maintenance mode) supports the bootstrap of MDM diagnostics.
- **R3000** (normal mode) supports the bootstrap of ULTRIX-32, as well as any user application image that conforms to the boot formats described in this section.

**DIFFERENCES:** *The KN210 system contains two console programs: maintenance mode (CVAX) and normal mode (R3000). See Section 3.7 for normal mode commands that you type at the >> prompt. Normal mode commands are case sensitive. See Section 3.9 for maintenance mode commands that you type at the >>> prompt.*

*Similar MicroVAX systems contain one CVAX console program.*

A KN210 bootstrap occurs under the following conditions:

- You type **BOOT** at the maintenance mode prompt (>>>).
- After bootpath is set, you type **boot** (lowercase letters) at the normal mode prompt (>>).

### 3.5.1 ULTRIX-32 Bootstrap

The ULTRIX-32 operating system is booted by the R3000 processor under one of the following conditions:

- The operation switch is set to the normal position and either of the following is true:
  - Autoboot is set to a.
  - You type boot at the normal mode prompt (>>).
- The operating system initiates a reboot operation.

You can use one of the following ports as the ULTRIX-32 boot device:

- KN210 I/O module Ethernet controller
- KN210 I/O module DSSI controller
- KN210 Q22-bus MSCP or TMSCP controller

Table 3-3 lists the supported ULTRIX-32 boot devices. The table correlates the boot device names expected in a boot command with the corresponding supported devices.

Boot device names consist of a two- or three-letter device code (letters a through z).

**Table 3-3: ULTRIX-32 Supported Boot Devices**

Device Type	Protocol	Number of Units Per Storage Adapter	Device Name
RF-series ISE	DSSI	7	rf
RA-series fixed-disk	MSCP	4	ra
Tape drive	TMSCP	1	tm
Ethernet adapter	MOP	1	mop

You boot the ULTRIX-32 operating system at the normal mode prompt (>>), using the commands explained in Section 3.6. Normal mode commands are case sensitive (see Section 3.7).

Boot the system as follows:

#### On Installation

1. On the H3602-AB CPU I/O panel, set the operation switch to the normal position (•).

2. Set the on/off power switch to 1 (on).
3. After the system has completed the power-up self-tests successfully, the normal mode prompt is displayed (>).

To define the bootpath, define the environment variable for the desired boot device, using lowercase letters, then boot the system to save the desired boot device. In Example 3-2, the boot device is an ISE at node 0.

### Example 3-2: Command Procedure to Boot on Installation

```
>> setenv bootpath rf(0,0,0) <vmunix>      !For file name, type
                                           !vmunix or
                                           !application file name.
>> boot
```

### On Power-Up of Existing System

1. On the H3602-AB CPU I/O panel, set the operation switch to the normal position (→).
2. Set the on/off power switch to on (1).
3. After the system has completed the power-up self-tests successfully, the R3000 processor attempts to boot the operating system through the previously defined boot device.

## 3.5.2 MDM Bootstrap

When you enter maintenance mode and type **BOOT MUxx** or **SET BOOT MUxx** at the >>> prompt, the CVAX processor boots the MDM operating system.

Before dispatching to the primary CVAX bootstrap (VMB), the KN210 CVAX processor firmware initializes the system to a known state, as follows:

1. Checks CPMBX<2>(RIP), bootstrap in progress. If it is set, bootstrap fails and the console displays the message **Failure.** in the selected console language.
2. Validates the boot device name. If none exists, supplies a list of available devices and issues a boot device prompt. If you do not specify a device within 30 seconds, uses **ESA0**.
3. Writes a form of this boot request, including active boot flags and boot device (**BOOT/R5:0 ESA0**, for example), to the console terminal.
4. Sets CPMBX<2>(BIP).

5. Initializes the Q22-bus scatter-gather map.
6. Validates the PFN bitmap. If invalid, rebuilds it.
7. Searches for a 128-Kbyte contiguous block of good memory as defined by the PFN bitmap. If 128 Kbytes cannot be found, the bootstrap fails.
8. Initializes the general purpose registers:
 

R0	Address of descriptor of the boot device name or 0 if none specified
R2	Length of PFN bitmap in bytes
R3	Address of PFN bitmap
R4	Time-of-day of bootstrap from PR\$_TODR
R5	Boot flags
R10	Halt PC value
R11	Halt PSL value (without halt code and mapenable)
AP	Halt code
SP	Base of 128-Kbyte good memory block plus 512
PC	Base of 128-Kbyte good memory block plus 512
R1, R6, R7, R8,	0
R9, FP	
9. Copies the virtual memory bootstrap (VMB) image from EPROM to local memory, beginning at the base of the 128 Kbytes of good memory block plus 512.
10. Exits from the firmware to VMB residing in memory.

Virtual Memory Bootstrap (VMB) is the primary MDM bootstrap. The KN210 VMB resides in the CVAX firmware and is copied into main memory before control is transferred to it. VMB then loads the secondary bootstrap image and transfers control to it.

Table 3-4 lists the supported R5 boot flags.

**Table 3-4: VMB Boot Flags**

Bit	Name	Description
0	RPB\$V_CONV	Conversational boot. At various points in the system boot procedure, the bootstrap code solicits parameters and other input from the console terminal.
1	RPB\$V_DEBUG	Debug. If this flag is set, the code for the XDELTA debugger is mapped into the system page tables of the running system.
2	RPB\$V_INIBPT	Initial breakpoint. If RPB\$V_DEBUG is set, the VMS operating system executes a BPT instruction in module INIT immediately after enabling mapping.
3	RPB\$V_BBLOCK	Secondary bootstrap from bootblock. When set, VMB reads logical block number 0 of the boot device and tests it for conformance with the bootblock format. If in conformance, the block is executed to continue the bootstrap. No attempt is made to perform a Files-11 bootstrap.
4	RPB\$V_DIAG	Diagnostic bootstrap. When set, the load image requested over the network is [SYS0.SYSMAINT]DIAGBOOT.EXE.
5	RPB\$V_BOOBPT	Bootstrap breakpoint. When set, a breakpoint instruction is executed in VMB and control is transferred to XDELTA before booting.
6	RPB\$V_HEADER	Image header. When set, VMB transfers control to the address specified by the file's image header. When not set, VMB transfers control to the first location of the load image.
8	RPB\$V_SOLICT	File name solicit. When set, VMB prompts the operator for the name of the application image file. The maximum file specification size is 17 characters.
9	RPB\$V_HALT	Halt before transfer. When set, VMB halts before transferring control to the application image.
31:28	RPB\$V_TOPSYS	This field can be any value from 0 through F. This flag changes the top-level directory name for system disks with multiple operating systems. For example, if TOPSYS is 1, the top-level directory name is [SYS1...].

Table 3-5 lists the supported MDM boot devices.

**Table 3-5: Supported MDM Boot Devices**

Boot Name	Controller Type	Device Type(s)
<b>Disk</b>		
[node\$]DIAn	On-board DSSI	RF-series ISEs
DUcn	RQDX3 MSCP	RD-series fixed disk drives, RX33, RX50
	KDA50 MSCP	RA-series fixed disk drives
	KFQSA MSCP	RF-series ISEs
	KLESI	RC25
DLcn	RLV12	RL01, RL02
<b>Tape</b>		
MUcn	TQK70 MSCP	TK70
KLESI	TU81E	-
<b>Network</b>		
ESA0	On-board Ethernet	-
XQcn	DELQA	-
	DESQA	-

### 3.5.3 MDM Restart

An MDM restart is the process of bringing up the MDM operating system from a known initialization state following a processor halt.

A restart occurs under the conditions listed in Table 3–2, earlier in this chapter.

To restart MDM, the firmware searches system memory for the Restart Parameter Block (RPB), a data structure constructed for this purpose by VMB. If the firmware finds a valid RPB, it passes control to the operating system at an address specified in the RPB.

The firmware keeps a RIP (restart-in-progress) flag in CPMBX, which it uses to avoid repeated attempts to restart a failing operating system. The operating system maintains an additional RIP flag in the RPB. The RPB is a page-aligned control block that can be identified by its signature in the first three longwords:

- +00 (first longword) = physical address of the RPB
- +04 (second longword) = physical address of the restart routine
- +08 (third longword) = checksum of first 31 longwords of restart routine

The firmware finds a valid RPB as follows:

1. Searches for a page of memory that contains its address in the first longword. If none is found, the search for a valid RPB has failed.
2. Reads the second longword in the page (the physical address of the restart routine). If it is not a valid physical address, or if it is zero, returns to step 1. The check for zero is necessary to ensure that a page of zeros does not pass the test for a valid RPB.
3. Calculates the 32-bit two's-complement sum (ignoring overflows) of the first 31 longwords of the restart routine. If the sum does not match the third longword of the RPB, returns to step 1.
4. If the sum matches, a valid RPB has been found.

## 3.6 Normal Mode Overview

When the KN210 is in normal mode, the console reads and interprets commands received on the console terminal at the normal mode prompt (**>>**).

This R3000 interactive command language allows you to make use of environment variables to pass information to the ULTRIX-32 operating system.

You can use normal mode commands to boot the ULTRIX-32 operating system, set up automatic booting, and deposit or examine I/O address space and memory.

### 3.6.1 Control Characters in Normal Mode

Table 3-6 lists the characters that have special meaning in normal mode.

**Table 3-6: Normal-Mode Control Characters**

Character	Action
<b>&lt;CR&gt;</b>	Ends a command line. Command characters are buffered until you press <b>Return</b> .
<b>Delete</b>	Deletes the previously typed character. If you define the console terminal as hard copy (environment variable <i>term</i> set to <i>hardcopy</i> ), the deleted text is displayed surrounded by backslashes. If the console terminal is a CRT (environment variable <i>term</i> set to <i>crt</i> ), each delete is displayed with the sequence <b>&lt;BS&gt;&lt;SP&gt;&lt;BS&gt;</b> . Deletes received are ignored when there are no characters to be deleted.
<b>Cn/C</b>	Causes the console to abort the processing of a command.
<b>Cn/O</b>	Causes console output to be discarded until you enter the next <b>Cn/O</b> or until the next console prompt or error message is issued. <b>Cn/O</b> is also canceled when you enter <b>Cn/C</b> .
<b>Cn/Q</b>	Resumes console output that was suspended when you entered <b>Cn/S</b> .
<b>Cn/R</b>	Causes the current command line to be displayed without any deleted characters.
<b>Cn/S</b>	Suspends output on the console terminal until you enter <b>Cn/O</b> .
<b>Cn/U</b>	Discards all characters accumulated for the current line.
<b>Cn/V</b>	Suppresses any special meaning associated with the next character.



## 3.6.2 Environment Variables in Normal Mode

The KN210 console makes use of environment variables to pass information to the operating system.

There are three types of variables

- Volatile (lost when power resumes)
- Nonvolatile (maintained after power resumes)
- Fixed (rebuilt when power is turned on)

You can define additional environment variables, but those you define will be lost when power is removed.

Table 3-7 lists the default variables.

**Table 3-7: Environment Variables**

Variable	Type	Description
<i>baud</i>	Fixed	The baud rate of the console terminal line is determined by the baud rate select switch inside the H3602-AB CPU I/O panel. The factory setting is 9600. Allowed values are 300, 600, 2400, 4800, 9600, 19,200, and 38,400.
<i>bitmap</i>	Fixed	Indicates the address of the memory bitmap. The bitmap keeps track of good and bad memory pages. Each bit corresponds to one page in memory; 1 indicates the page is good, and 0 indicates the page is bad.
<i>bitmaplen</i>	Fixed	Indicates the length of the memory bitmap in bytes.
<i>bootpath</i>	Nonvolatile	Indicates the default bootpath. The system uses this variable when you type the auto command. An example of a bootpath definition is: <code>rf0,0,0vmmunix</code> .
<i>bootmode</i>	Nonvolatile	Determines what programs run when the system is turned on or reset. Use one of the following codes:  a    Autoboots the operating system using the <i>bootpath</i> variable.  d    Halts the system after performing power-up diagnostics.
<i>console</i>	Fixed	The system always selects TTY(0) as the console device.
<i>osconsole</i>	Fixed	The system always selects TTY(0) as the console device.

**Table 3-7 (Cont.): Environment Variables**

Variable	Type	Description
<i>svctype</i>	Fixed	Contains information used to identify the processor. Bits 24:31 contain the CPU type. Bits 16:23 contain the system type (six for KN210). Bits 8:15 contain the firmware revision level. Bits 0:7 contain the hardware version level.

### 3.7 Normal Mode Commands

The R3000 console program displays the normal mode prompt (>>) when it is ready to accept commands.

Table 3-8 lists the supported normal mode console commands.

**Table 3-8: KN210 Normal Mode Commands**

Command	Description
<b>boot</b>	Boots the operating system
<b>continue</b>	Returns control to the processes interrupted by a halt signal
<b>d</b>	Deposits data at a given address
<b>dump</b>	Dumps memory to the screen
<b>e</b>	Examines memory
<b>fill</b>	Deposits data in an address range
<b>go</b>	Resumes execution of the program in memory
<b>help</b>	Displays the syntax of console commands
<b>?</b>	Displays the syntax of console commands
<b>init</b>	Reinitializes memory
<b>maint</b>	Causes the console to enter maintenance mode
<b>printenv</b>	Displays console environment variables
<b>setenv</b>	Sets console environment variables
<b>unsetenv</b>	Unsets console environment variables

Observe the following rules when you type normal mode commands:

- All commands typed at normal mode level are case sensitive with respect to parsing commands; case is preserved when you assign values to environment variables.
- Type all normal mode console commands, using only ASCII characters. Values that you enter for environment variables may contain any 8-bit character code.
- Command execution begins when you press **Return**.
- Enter numeric values as follows:
  - Enter *decimal values* as a string of decimal digits with no leading zeros (for example, 123).
  - Enter *octal values* as a string of octal digits with a leading zero (for example, 0177).
  - Enter *hexadecimal values* as a string of hexadecimal digits preceded by 0x (for example, 0x3ff).
  - Enter *binary values* as a string of binary digits preceded by 0b (for example, 0b1001).
- When reading or writing to memory, you have a choice of data sizes: byte, halfword, or word. Leading zeros are dropped.

Because a word is 4 bytes, successive addresses, when referenced by a word, are successive multiples of 4. For example, the address following 0x80000004 is 0x80000008. An error occurs if you try to specify an address that is not on a boundary for the data size you are using.

### 3.7.1 Conventions Used in This Section

- Letters are to be typed exactly as they appear.
- Letters in italics represent arguments for which you supply values. (Note that the Help and menu screens display these arguments in all capital letters.)
- Arguments enclosed in square brackets ( [ ] ) are optional.
- Ellipses ( ... ) follow an argument that can be repeated.
- A vertical bar ( | ) separates choices.
- Parentheses are used as in algebraic expressions. For example, the following sequence means enter -b or -h or -w:

- (b|h|w)

### 3.7.2 Getting Help

You can get help with console command syntax in several ways:

- You can type the word *help* or a question mark ( ? ) to display a menu of all console commands.
- You can enter the name of the command for which you want help as an argument to *help* or as a question mark ( ? ).

For example, entering ? e at the console prompt ( >> ) displays the syntax for the examine ( e ) command:

```
e [- (b|h|w)] ADDR
>>
```

- If you type an incorrect command line, you get a Help screen.

For example, the e command requires an *addr* argument. If you type e -b at the console prompt ( >> ) without entering an address, the screen will display the correct syntax for the command:

```
e [- (b|h|w)] ADDR
>>
```

### 3.7.3 boot

The boot command loads the file that contains the operating system.

*Format:*

**boot [-f *file*] [-s] [-n] [*arg...*]**

The optional -f flag followed by the *file* parameter specifies the file you want to use during a boot procedure. If you do not specify the -f flag and a file, the file specified by the environment variable *bootpath* is loaded.

The *file* parameter has the format.

**dev([*controller*][,*unit-number*] [,*partition-number*])(*filename*)**

- *dev* indicates the device from which you are booting the operating system. Typical devices are *rf* for RF-series ISEs, *ra* for RA-series hard disk drives, *tm* for a tape, and *mop* for a network. Typing *mop* nullifies the other arguments in the list. Table 3-9 lists the device names for each device.

**Table 3-9: Boot Device Names (Normal Mode)**

Device Type	Protocol	Number of Units Per Storage Adapter	Device Name
RF-series ISE	DSSI	7	<i>rf</i>
RA-series fixed-disk	MSCP	4	<i>ra</i>
Tape drive	TMSCP	1	<i>tm</i>
Ethernet adapter	MOP	1	<i>mop</i>

- *controller* indicates the ID number of the controller for the device from which you are booting the operating system.
- *unit-number* indicates the unit number of the device from which you are booting the operating system.

To display a list of devices, their unit numbers, and controller numbers, enter maintenance mode and issue the command **SHOW DEVICE** at the maintenance prompt **>>>**. After viewing the display, type **EXIT** and press **[Return]** to return to the normal console prompt **>>**.

**Example:**

```
>> maint
>>> SHOW DEVICE
DSSI Node 0 (R3WB0A)
-DIA0 -rf(0,0,*) (RF71)

DSSI Node 1 (R3QDVA)
-DIA1 -rf(1,1,*) (RF71)

DSSI Node 7 (*)

UQSSP Tape Controller 0 (774500)
-MUA0 -tm(0,0) (TK70)

Ethernet Adapter
-ESA0 -se -mop() (08-00-2B-0C-C4-75)

>>> EXIT
>>
```

As in the preceding example, the **SHOW DEVICE** command displays the CVAX device names and the R3000 device names, which are followed by the controller number and unit number in parentheses. The asterisk indicates the *partition-number* variable, which is determined during installation of the operating system software.

- *partition-number* indicates the number (or other designator) of the partition from which you are booting the operating system. When you are booting from a tape, this number is not used because the boot file must be the first file on the tape. When you are booting from a disk, this number depends on how you partitioned the disk when you installed your operating system software. Refer to your software installation manual if you need a reminder about disk partition indicators.
- *filename* indicates the name of the operating system file.

The optional **-s** flag causes the operating system to boot in single-user mode. Unless **-s** is specified, the system will boot in multiuser mode.

The optional **-n** flag causes the specified file to be loaded but not executed.

The optional **org** parameter contains any information to be passed to the booted image.

***Examples:***

```
>> boot -f ra(0,0,0)vmunix
```

**This command boots the file vmunix, located in the (a) partition of the first hard disk (unit number 0), using controller 0.**

```
>> boot -f rf(2,2,c)vmunix
```

**This command boots the file vmunix, located in the c partition of the second RF-series ISE (unit number 2), using controller 2.**

```
>> boot -f tm(0,5)
```

**This command boots from the tape, which is unit 5 and controller 0.**

### 3.7.4 continue

**CAUTION:** *If the operating system state has not been properly saved (halted), do not type continue. Doing so may cause the processor to hang.*

The continue command returns control to the processes interrupted by a halt signal. Use this command if you inadvertently halt the system by pressing **Break** or the Halt button.

*Format:*

**continue**



### 3.7.5 d (deposit)

The **d** (deposit) command deposits a single byte, halfword, or word value at the specified address.

**NOTE:** The R3000 halfword consists of 16 bits, and the word consists of 32 bits.

*Format:*

**d *[-(b | h | w)] addr val***

The first parameter, which is optional, indicates the data size. If not given, data size defaults to word. If you do not specify a data size, a word is used.

- **-b** deposits 1 byte of data.
- **-h** deposits a halfword (2 bytes) of data.
- **-w** deposits a word (4 bytes) of data.

The *addr* parameter indicates the address to which you want data written. System address space is in the range 0x80000000 to 0xbf000000.

The *val* parameter contains the data you want deposited at the given address.

*Example:*

```
>> d -w 0x80000000 0xffffffff
```

This command deposits the value 0xffffffff, with a data size of one word, at address 0x80000000.

### 3.7.6 dump

This command shows a formatted display of the contents of memory.

*Format:*

**dump** **[-(b|h|w)]** **[-(o|d|u|x|c|B)]** *rng*

The first parameter, which is optional, indicates the data size. If not given, data size defaults to word. If you do not specify a data size, the system uses a word.

- **-b** displays memory in bytes.
- **-h** displays memory in halfwords.
- **-w** displays memory in words.

The next parameter, also optional, determines how data is displayed.

- **-o** displays memory in octal format.
- **-d** displays memory in decimal format.
- **-u** displays memory in unsigned decimal format.
- **-x** displays memory in hexadecimal format.
- **-c** displays memory in ASCII format.
- **-B** displays memory in binary format.

If no format argument is given, hexadecimal format is used.

The *rng* parameter indicates the range of memory you want to see. You can specify the range in one of two ways:

- *addr#cnt* displays the number of addresses specified by *cnt*, beginning at *addr*.
- *addr:addr* displays all values between the specified addresses.

***Examples:***

```
>> dump 0x80000000#0xf
```

This command uses hexadecimal format to dump the first 15 words of memory to the screen.

```
>> dump -b 0x80000000#0xf
```

This command uses hexadecimal format to dump the first 15 bytes of memory to the screen.

The dump display shows rows of address contents. The leftmost column gives the address of the first field in each row.

### 3.7.7 e (examine)

The **e** (examine) command examines the byte, halfword, or word at the specified address.

**NOTE:** *The R3000 halfword consists of 16 bits, and the word consists of 32 bits.*

*Format:*

**e** **[-(b | h | w)]** *addr*

The first parameter, which is optional, indicates the data size. If not given, data size defaults to word. If you do not specify the data size, a word is used.

- **-b** indicates a single byte.
- **-h** indicates a halfword.
- **-w** indicates a word.

The *addr* parameter indicates an address in the range 0x80000000 to 0xbfffffff.

When you enter the examine command, a display similar to the following appears:

```
0x80000005:  65 0x41    'A'
```

The left-hand field echoes the address you entered.

The next three fields display the contents of the address in decimal, hexadecimal, and ASCII formats, respectively. If the ASCII character is unprintable, it is displayed as an octal value preceded by a backslash: for example, '\032'.

*Example:*

```
>> e 0x80000000
```

This command examines the word at address 0x80000000. The resulting display might look like this:

```
0x80000000:      1008385985      0x3c1abfc1      '\301'
```

### 3.7.8 fill

The fill command writes a specified value to a range of memory. If you do not specify a value, the system puts zeros in the memory range.

*Format:*

**fill *[-(b | h | w)] [-v val] rng***

The first parameter, which is optional, indicates the data size. If not given, data size defaults to word.

- **-b** indicates bytes.
- **-h** indicates halfwords.
- **-w** indicates words.

The optional parameter **-v val** specifies the numeric value to write to memory. If you do not specify a value, all zeros are written. If the size of *val* does not match the data size parameter, *val* is truncated or expanded as necessary.

The *rng* parameter indicates the memory range. You can specify the range in one of two ways:

- **addr#cnt** fills addresses beginning at *addr* and continuing for *cnt* locations.
- **addr:addr** fills all locations between the two given addresses.

*Example:*

```
>> fill -v 0xffffffff 0x80000010:0x800000ff
```

This command sets all bits to 1 at addresses 16 to 255.

### **3.7.9 go**

The **go** command transfers control to the indicated entry-point address.

*Format:*

**go** [*pc*]

The optional *pc* parameter indicates the entry-point address you want to use.

If you do not specify an entry address, the system uses the entry point of the program module that was most recently loaded. If no program module was previously loaded, the system uses 0 as the entry-point address.

### 3.7.10 help

The help command displays the correct syntax for the console commands.

*Format:*

**help [cmd]**

The optional *cmd* parameter indicates the command for which you want information. If you do not specify *cmd*, the complete console menu appears.

*Example:*

```
>> help
CMD:
  boot [-f FILE] [-s] [-n] [ARG...]
  continue
  d [-(b|h|w)] ADDR VAL
  dump [-(b|h|w)] [-(o|d|u|x|c|B)] RNG
  e [-(b|h|w)] ADDR
  fill [-(b|h|w)] [-v VAL] RNG
  go [PC]
  help [CMD]
  init
  maint
  printenv [EVAR...]
  setenv EVAR STR
  unsetenv EVAR
  ? [CMD]
RNG:
  ADDR#CNT
  ADDR:ADDR
```

### **3.7.11 ?**

The ? command functions exactly like the help command (Section 3.7.10).

*Format:*

**? [cmd]**



### **3.7.12 init**

The init command fully initializes the system.

*Format:*

**init**

The system performs the following:

- Clears memory
- Resets I/O
- Initializes all supported devices

The effect of the init command is identical to turning the power on or pressing the Reset button, except that the system does not execute its self-test. The memory size and Ethernet address are displayed.

*Example:*

```
>> init
Memory Size: 16777216 (0x1000000) bytes
Ethernet Address: 08-00-2b-0c-c5-f6
```

### 3.7.13 printenv

The `printenv` command displays the current value for the specified environment variable.

*Format:*

`printenv [env...]`

The optional `env` parameter indicates the variable whose value you want to see. If you do not specify a variable, the complete environment variable table appears.

*Examples:*

```
>> printenv
bootpath=rf(0,0,0)vmunix
bootmode=*
console=0
baud=9600
systype=0x82060102
bitmap=0xa000fcc0
bitmaplen=0xc0
osconsole=0
```

```
>> printenv bootpath
bootpath=rf(0,0,0)vmunix
```

### 3.7.14 setenv

The **setenv** command assigns new values to the specified environment variable. Refer to the discussion of the **printenv** command (Section 3.7.13) for a description of each variable.

*Format:*

**setenv** *var str*

- The *var* parameter indicates the variable you want to set.
- The *str* parameter indicates the value you want to specify.

*Example:*

```
>> setenv bootmode a
```

The command in the example above assigns a value of "a" to the *bootmode* variable. This will cause the system to autoboot at power-up.

You can also add your own environment variables, as explained in Section 3.7. These variables are stored in volatile memory. The environment variables table can contain up to 16 variables, for a total of 256 characters.

### 3.7.15 unsetenv

The `unsetenv` command removes the specified variable from the environment variables table.

*Format:*

`unsetenv env`

The *env* parameter indicates the variable you are removing. Refer to Table 3-7 earlier in this section for a description of each variable.

The `unsetenv` command does not affect the environment variables stored in nonvolatile memory. These variables are reset at the next reset or power cycle.

## 3.8 Maintenance Mode Overview

Whenever the CVAX console program is running, the maintenance mode prompt (>>>) is displayed on the console terminal and the KN210 is halted as described in Section 3.3.

In maintenance mode, you can examine and alter the state of the processor by typing certain console commands and characters. Table 3-10 lists the keypad control characters that have special meaning in maintenance mode.

**Table 3-10: KN210 Console Control Characters (Maintenance Mode)**

Character	Action
<b>Return</b>	Also <CR>. The carriage return ends a command line. No action is taken on a command until after it is terminated by a carriage return. A null line terminated by a carriage return is treated as a valid, null command. No action is taken, and the console prompts for input. Carriage return is echoed as carriage return, line feed (<CR><LF>).
<b>X (Rubout)</b>	<p>When you press the <b>X</b> (rubout) key, the console deletes the previously typed character. The resulting display differs, depending on whether the console is a video or a hard-copy terminal.</p> <p>For hard-copy terminals, the console echoes a backslash (\), followed by the character being deleted. If you press additional rubouts, the additional deleted characters are echoed. If you type a nonrubout character, the console echoes another backslash, followed by the character typed. The result is to echo the characters deleted, surrounding them with backslashes. For example:</p> <p>EXAMI;E<b>X</b> (rubout) <b>X</b> (rubout) NE&lt;CR&gt;</p> <p>The console echoes: EXAMI;E\ E;\ NE&lt;CR&gt; The console sees the command line: EXAMINE&lt;CR&gt; For video terminals, the previous character is erased and the cursor is restored to its previous position. The console does not delete characters past the beginning of a command line. If you press more rubouts than there are characters on the line, the extra rubouts are ignored. A rubout entered on a blank line is ignored.</p>
<b>CW/U</b>	Echoes ^U<CR>, and deletes the entire line. Entered but otherwise ignored if typed on an empty line.
<b>CW/S</b>	Stops output to the console terminal until <b>CW/Q</b> is typed. Not echoed.
<b>CW/O</b>	Resumes output to the console terminal. Not echoed.
<b>CW/R</b>	Echoes <CR><LF>, followed by the current command line. Can be used to improve the readability of a command line that has been heavily edited.
<b>CW/C</b>	Echoes ^C<CR> and aborts processing of a command. When entered as part of a command line, deletes the line.

**Table 3-10 (Cont.): KN210 Console Control Characters (Maintenance Mode)**

Character	Action
<b>CWO</b>	Ignores transmissions to the console terminal until the next <b>CWO</b> is entered. Echoes ^O when disabling output, not echoed when it reenables output. Output is reenabled if the console prints an error message, or if it prompts for a command from the terminal. Output is also enabled by entering maintenance mode, by pressing the <b>Break</b> key, and by pressing <b>CWC</b> .

### 3.8.1 Command Syntax in Maintenance Mode

The console accepts commands up to 80 characters long. Longer commands produce error messages. The character count does not include rubouts, rubbed-out characters, or the **Return** at the end of the command.

You can abbreviate a command by entering only as many characters as are required to make the command unique. Most commands can be recognized from their first character. See Table 3-14.

The console treats two or more consecutive spaces and tabs as a single space. Leading and trailing spaces and tabs are ignored. You can place command qualifiers after the command keyword or after any symbol or number in the command.

All numbers (addresses, data, counts) are hexadecimal (hex) except for GPR symbolic names, which are in decimal. The hex digits are 0 through 9 and A through F. You can use uppercase and lowercase letters in hex numbers (A through F) and commands.

The following symbols are qualifier and argument conventions:

- [ ] an optional qualifier or argument
- | | a required qualifier or argument

### 3.8.2 Address Specifiers in Maintenance Mode

Several commands take an address or addresses as arguments. An address defines the address space and the offset into that space. The console supports six address spaces:

Physical memory	General purpose registers (GPRs)
Virtual memory	Internal processor registers (IPRs)
Protected memory	The PSL

The address space that the console references is inherited from the previous console reference, unless you explicitly specify another address space. The initial address space is physical memory.

### 3.8.3 Symbolic Addresses in Maintenance Mode

The console supports symbolic references to addresses. A symbolic reference defines the address space and the offset into that space. Table 3–11 lists symbolic references supported by the console, grouped according to address space. You do not have to use an address space qualifier when using a symbolic address.

**Table 3–11: Console Symbolic Addresses (Maintenance Mode)**

Symbol	Address	Symbol	Address
<b>GPR Address Space (/G)</b>			
R0	0	R1	1
R2	2	R3	3
R4	4	R5	5
R6	6	R7	7
R8	8	R9	9
R10	0A	R11	0B
R12	0C	R13	0D
R14	0E	R15	0F
AP	0C	FP	0D
SP	0D	PC	0E
PSL	–	–	–
<b>IPR Address Space (/I)</b>			
pr\$_ksp	00	pr\$_esp	01
pr\$_esp	02	pr\$_usp	03
pr\$_isp	04	pr\$_p0br	08
pr\$_p0lr	09	pr\$_p1br	0A
pr\$_p1lr	0B	pr\$_sbr	0C
pr\$_slr	0D	pr\$_pcbb	10
pr\$_ecbb	11	pr\$_ipl	12
pr\$_astlv	13	pr\$_sirr	14
pr\$_sirr	15	pr\$_iccr	18
pr\$_nicr	19	pr\$_icr	1A
pr\$_todr	1B	pr\$_rxcs	20
pr\$_rxdbr	21	pr\$_txcs	22

**Table 3-11 (Cont.): Console Symbolic Addresses (Maintenance Mode)**

<b>Symbol</b>	<b>Address</b>	<b>Symbol</b>	<b>Address</b>
<b>IPR Address Space (/I)</b>			
pr\$_txdb	23	pr\$_tldr	24
pr\$_cadr	25	pr\$_mcsr	26
pr\$_mser	27	pr\$_savpc	2A
pr\$_savpal	2B	pr\$_ioreset	37
pr\$_mapen	38	pr\$_tbis	39
pr\$_tbis	3A	pr\$_sid	3E
pr\$_tbchk	3F	-	-
<b>Physical Memory (/P)</b>			
qbio	20000000	qbmern	30000000
qbmbn	20080010	-	-
rom	20040000	-	-
cacr	20084000	bdr	20084004
dacr	20080000	dsr	20080004
dmear	20080008	dsar	2008000C
ipcr0	20001f40	ipcr1	20001f42
ipcr2	20001f44	ipcr3	20001f46
sec_rsm	20140400	sec_cr	20140010
sec_cdal	20140020	sec_dledr	20140030
sec_ad0mat	20140130	sec_sj0mak	20140134
sec_ad1mat	20140140	sec_ad1mak	20140144
sec_tcr0	20140100	sec_tir0	20140104
sec_tnir0	20140108	sec_tivr0	2014010c
sec_tcr1	20140110	sec_tir1	20140114
sec_tnir1	20140118	sec_tivr1	2014011c
memcar0	20080100	memcar1	20080104
memcar2	20080108	memcar3	2008010c
memcar4	20080110	memcar5	20080114
memcar6	0080118	memcar7	2008011c
memcar8	20080120	memcar9	20080124
memcar10	20080128	memcar11	2008012c
memcar12	20080130	memcar13	20080134
memcar14	20080138	memcar15	2008013c
memcar16	20080140	memcar17	20080144
nisarom	20084200	nirdp	20084400
nirap	20084404	nibuf	20120000



**Table 3-11 (Cont.): Console Symbolic Addresses (Maintenance Mode)**

Symbol	Address	Symbol	Address
<b>Physical Memory (/P)</b>			
msi_sbb	20084600	msi_sc1	20084604
msi_sc2	20084608	msi_csr	2008460C
msi_id	20084610	msi_slcar	20084614
msi_destat	20084618	msi_dstmo	2008461C
msi_data	20084620	msi_dmctrl	20084624
msi_cmlotc	20084628	msi_dmaddr1	2008462C
msi_dmaddrh	2008463C	msi_dmabyte	20084634
msi_stlp	20084638	msi_lt1p	2008463C
msi_ilp	20084640	msi_dactrf	20084644
msi_cstat	20084648	msi_dstat	2008464C
msi_comm	20084650	msi_dictrl	20084654
msi_clock	20084658	msi_bhdiag	2008465C
msi_sidiag	20084660	msi_dmdiag	20084664
msi_mcdiag	20084668	msi_ram	20100000

Table 3-12 lists symbolic addresses that you can use in any address space.

**Table 3-12: Symbolic Addresses Used In Any Address Space**

<b>Symbol</b>	<b>Description</b>
<b>*</b>	The location last referenced in an EXAMINE or DEPOSIT command.
<b>+</b>	The location immediately following the last location referenced in an EXAMINE or DEPOSIT command. For references to physical or virtual memory spaces, the location referenced is the last address, plus the size of the last reference (1 for byte, 2 for word, 4 for longword, 8 for quadword). For other address spaces, the address is the last address referenced plus one.
<b>-</b>	The location immediately preceding the last location referenced in an EXAMINE or DEPOSIT command. For references to physical or virtual memory spaces, the location referenced is the last address minus the size of this reference (1 for byte, 2 for word, 4 for longword, 8 for quadword). For other address spaces, the address is the last address referenced minus one.
<b>●</b>	The location addressed by the last location referenced in an EXAMINE or DEPOSIT command.

### **3.8.4 Command Qualifiers in Maintenance Mode**

You can enter console command qualifiers in any order on the command line after the command keyword. There are three types of qualifiers: data control, address space control, and command specific. Table 3-13 lists and describes the data control and address space control qualifiers. Command specific qualifiers are listed in the descriptions of individual commands.

**Table 3–13: Console Command Qualifiers (Maintenance Mode)**

<b>Qualifier</b>	<b>Description</b>
<b>Data Control</b>	
<b>/B</b>	The data size is byte.
<b>/W</b>	The data size is word.
<b>/L</b>	The data size is longword.
<b>/Q</b>	The data size is quadword.
<b>/N:(count)</b>	An unsigned hexadecimal integer that is evaluated into a longword. This qualifier determines the number of additional operations that are to take place on EXAMINE, DEPOSIT, MOVE, and SEARCH commands. An error message appears if the number overflows 32 bits.
<b>/STEP:(size)</b>	Step. Overrides the default increment of the console current reference. Commands that manipulate memory, such as EXAMINE, DEPOSIT, MOVE, and SEARCH, normally increment the console current reference by the size of the data being used.
<b>/WRONG</b>	Wrong. Used to override or set error bits when referencing main memory. On writes, uses the complement. On reads, ignores ECC errors.
<b>Address Space Control</b>	
<b>/G</b>	General purpose register (GPR) address space, R0–R15. The data size is always longword.
<b>/I</b>	Internal processor register (IPR) address space. Accessible only by the MTPR and MFPR instructions. The data size is always longword.
<b>/V</b>	Virtual memory address space. All access and protection checking occur. If access to a program running with the current PSL is not allowed, the console issues an error message. Deposits to virtual space cause the PTE<M> bit to be set. If memory mapping is not enabled, virtual addresses are equal to physical addresses. Note that when you examine virtual memory, the address space and address in the response is the physical address of the virtual address.
<b>/P</b>	Physical memory address space.
<b>/M</b>	Processor status longword (PSL) address space. The data size is always longword.
<b>/U</b>	Access to console private memory is allowed. This qualifier also disables virtual address protection checks. On virtual address writes, the PTE<M> bit is not set if the /U qualifier is present. This qualifier is not inherited; it must be respecified on each command.

### 3.8.5 Maintenance Mode Command Keywords

Table 3-14 lists maintenance mode command keywords by type. Table 3-15 lists the parameters, qualifiers, and arguments for each console command. Parameters, used with the SET and SHOW commands only, are listed in the first column along with the command.

Although it is possible to abbreviate by using the minimum number of characters required to uniquely identify a command or parameter, these abbreviations may become ambiguous at a later time if a new command or parameter is added in an updated version of the firmware. For this reason, you should not use abbreviations in programs.

**Table 3-14: Command Keywords by Type (Maintenance Mode)**

Processor Control	Data Transfer	Console Control
BOOT	EXAMINE	CONFIGURE
CONTINUE	DEPOSIT	FIND
HALT	MOVE	REPEAT
INITIALIZE	SEARCH	SET
NEXT	X	SHOW
START		TEST
UNJAM		!
		EXIT

**Table 3-15: Console Command Summary (Maintenance Mode)**

Command	Qualifiers <sup>1</sup>	Argument	Other(s)
BOOT	/R5:(boot_flags) or /(boot_flag+)	[(boot_device)]	-
CONFIGURE	-	-	-
CONTINUE	-	-	-
DEPOSIT	/B /W /L /Q /G /I /N /P /M /U /N:(count) /STEP:(size) /WRONG [(mask)]	(address)	(data) [(data)]
EXAMINE	/B /W /L /Q /G /I /N /P /M /U /N:(count) /STEP:(size) /WRONG/INSTRUCTION	[address]	-
EXIT	-	-	-
FIND	/MEM /RPB	-	-
HALT	-	-	-

<sup>1</sup> [ ] denotes a mandatory item that must be syntactically correct.

[ ] denotes an optional item.

**Table 3-15 (Cont.): Console Command Summary (Maintenance Mode)**

Command	Qualifiers <sup>1</sup>	Argument	Other(s)
HELP	-	-	-
INITIALIZE	-	-	-
MOVE	/B /W /L /Q /N /P /U /N (count) /STEP (size) /WRONG [(mask)]	(src_address)	(dest_address)
NEXT	-	(count)	-
REPEAT	-	(command)	-
SEARCH	/B /W /L /Q /N /P /U /N (count) /STEP (size) /WRONG/NOT	(start_address)	(pattern) [(mask)]
SET BFLAG	-	(boot_flags)	-
SET BOOT	-	(device_string)	-
SET HOST	/DUP (/DSSI n /UQSSP) /DISK n /TAPE n ccr_address /MAINTENANCE /UQSSP /SERVICE n ccr_address	(node) n (controller_number)	[(task)]
SET LANGUAGE	-	(language_type)	-
SHOW BFLAG	-	-	-
SHOW BOOT	-	-	-
SHOW DEVICE	-	-	-
SHOW DSSI	-	-	-
SHOW ETHERNET	-	-	-
SHOW LANGUAGE	-	-	-
SHOW MEMORY	/FULL	-	-
SHOW QBUS	-	-	-
SHOW RLV12	-	-	-
SHOW UQSSP	-	-	-
SHOW VERSION	-	-	-
START	-	[(address)]	-
TEST	-	(test_number)	[(parameters)]
UNJAM	-	-	-
X	-	(address)	(count)

<sup>1</sup> | denotes a mandatory item that must be syntactically correct.

| denotes an optional item.

## 3.9 Maintenance Mode Commands

This section describes the maintenance mode commands. Enter the commands at the maintenance mode prompt (>>>). These commands may be typed in uppercase or lowercase letters. However, they are shown in all uppercase letters in this document to differentiate them from the normal mode commands, which must be typed in all lowercase letters.

### 3.9.1 BOOT

The **BOOT** command initializes the processor and transfers execution to VMB. VMB attempts to boot MDM from the specified device, or from the default boot device if none is specified. The console qualifies the bootstrap operation by passing a boot flags bitmap to VMB in R5.

*Format:*

**BOOT [qualifier-list] [device\_name]**

If you do not enter either the qualifier or the device name, the default value is used. Explicitly stating the boot flags or the boot device overrides, but does not permanently change, the corresponding default value.

Set the default boot device and boot flags with the **SET BOOT** and **SET BFLAG** commands. If you do not set a default boot device, the processor times out after 30 seconds and attempts to boot from the on-board Ethernet port, **ESA0**.

*Qualifiers:*

*Command specific:*

<b>/R5:(bitmap)</b>	A 32-bit hex value passed to VMB in R5. The console does not interpret this value. Use the <b>SET BFLAG</b> command to specify a default boot flags longword. Use the <b>SHOW BFLAG</b> command to display the longword. Table 3-4 lists the supported R5 boot flags.
<b>/(bitmap)</b>	Same as <b>/R5:(bitmap)</b>
<b>[device_name]</b>	A character string of up to 39 characters. Longer strings cause a <b>VAI, TOO BIG</b> error message. Apart from length, the console makes no attempt to interpret or validate the device name. The console converts the string to uppercase, then passes to VMB a string descriptor to this device name in R0.

*Example:*

```
>>> BOOT XQA0           ! Boot using default boot flags and
(BOOT/R5:10 DUA0)       ! specified device.
  2..
-XQA0
```

### **3.9.2 CONFIGURE**

The **CONFIGURE** command invokes an interactive mode that permits you to enter Q22-bus device names, then generates a table of Q22-bus I/O page device CSR addresses and interrupt vectors. **CONFIGURE** is similar to the VMS SYSGEN CONFIG utility. This command simplifies field configuration by providing information that is typically available only with a running operating system. Refer to the example below and use the **CONFIGURE** command as follows:

1. Enter **CONFIGURE** at the maintenance mode prompt (**>>>**).
2. Enter **HELP** at the **Device, Number?** prompt to see a list of devices whose CSR addresses and interrupt vectors can be determined.
3. Enter the device names and number of devices.
4. Enter **EXIT** to obtain the CSR address and interrupt vector assignments.

The devices listed in the **HELP** display are not necessarily supported by the KN210-AA CPU.

*Format:*

**CONFIGURE**

### Example:

>>> CONFIGURE

Enter device configuration, HELP, or EXIT

Device,Number? help

Devices:

LPV11	KXJ11	DLV11J	DZQ11	DZV11	DFA01
RLV12	TSV05	RXV21	DRV11W	DRV11B	DPV11
DMV11	DELQA	DEQNA	DESQA	RQDX3	KDA50
RRD50	RQC25	KFQSA-DISK	TQK50	TQK70	TU81E
RV20	KFQSA-TAPE	KMV11	IEQ11	DHQ11	DHV11
CXA16	CXB16	CXY08	VCB01	QVSS	LVN11
LVN21	QPSS	DSV11	ADV11C	AAV11C	AXV11C
KWV11C	ADV11D	AAV11D	VCB02	QDSS	DRV11J
DRQ3B	VSV21	IBQ01	IDV11A	IDV11B	IDV11C
IDV11D	IAV11A	IAV11B	MIRA	ADQ32	DTC04
DESN	IGQ11				

Numbers:

1 to 255, default is 1

Device,Number? tqk70

Device,Number? cxa16,2

Device,Number? dtc04

Device,Number? lpv11

Device,Number? exit

Address/Vector Assignments

-777514/200 LPV11

-774500/260 TQK70

-760440/300 CXA16

-760460/310 CXA16

-761242/320 DTC04

>>>



### 3.9.3 CONTINUE

**CAUTION:** *If the operating system state has not been properly saved (halted), do not type CONTINUE. Doing so may cause the processor to hang.*

The CONTINUE command causes the processor to begin instruction execution at the address currently contained in the PC. It does not perform a processor initialization. The console enters maintenance mode.

*Format:*

**CONTINUE**

*Example:*

>>> CONTINUE

### 3.9.4 DEPOSIT

The DEPOSIT command deposits data into the address specified. If you do not specify an address space or data size qualifier, the console uses the last address space and data size used in a DEPOSIT, EXAMINE, MOVE, or SEARCH command. After processor initialization, the default address space is physical memory, the default data size is longword, and the default address is zero. If you specify conflicting address space or data sizes, the console ignores the command and issues an error message.

#### *Format:*

**DEPOSIT [qualifier\_list] (address) (data) [data...]**

#### *Qualifiers:*

**Data control:** /B, /W, /L, /Q, /N:(count), /STEP:(size), /WRONG

**Address space control:** /G, /I, /P, /V, /U, /M

#### *Arguments:*

- (address)    A longword address that specifies the first location into which data is deposited. The address can be an actual address or a symbolic address.
- (data)       The data to be deposited. If the specified data is larger than the deposit data size, the firmware ignores the command and issues an error response. If the specified data is smaller than the deposit data size, it is extended on the left with zeros.
- (data)       Additional data to be deposited (as many as can fit on the command line).

#### *Examples:*

```
>>> D/P/B/N:1FF 0 0            ! Clear first 512 bytes of physical memory.
>>> D/V/L/N:3 1234 5            ! Deposit 5 into four longwords starting
                                 ! at virtual memory address 1234.
>>> D/N:8 R0 FFFFFFFF          ! Loads GPRs R0 through R8 with -1.
>>> D/N:200 - 0                 ! Starting at previous address, clear 512
                                 ! bytes.
>>> D/L/P/N:10/S:200 0 8        ! Deposit 8 in the first longword of
                                 ! the first 17 pages in physical
                                 ! memory starting at location 0.
```

### 3.9.5 EXAMINE

The **EXAMINE** command examines the contents of the memory location or register specified by the address. If no address is specified, + is assumed. The display line consists of a single character address specifier, the physical address to be examined, and the examined data.

**EXAMINE** uses the same qualifiers as **DEPOSIT**. However, the **/WRONG** qualifier causes examines to ignore ECC errors on reads from physical memory. The **EXAMINE** command also supports an **/INSTRUCTION** qualifier, which will disassemble the instructions at the current address.

#### *Format:*

**EXAMINE** [qualifier\_list] [address]

#### *Qualifiers:*

*Data control:* **/B**, **/W**, **/L**, **/Q**, **/N:(count)**, **/STEP:(size)**, **/WRONG**

*Address space control:* **/G**, **/I**, **/P**, **/V**, **/U**, **/M**

#### *Command specific:*

**/INSTRUCTION**    Disassembles and displays the VAX MACRO-32 instruction at the specified address.

#### *Arguments:*

[address]            A longword address that specifies the first location to be examined. The address can be an actual or a symbolic address. If no address is specified, + is assumed

### Examples:

```
>>> EXAMINE PC                                ! Examine the PC.
      G 0000000F FFFFFFFFC
>>> EXAMINE SP                                ! Examine the SP.
      G 0000000E 00000200
>>> EXAMINE PSL                               ! Examine the PSL.
      M 00000000 041F0000
>>> EXAMINE/M                                ! Examine PSL another way.
      M 00000000 041F0000
>>> EXAMINE R4/N:5                           ! Examine R4 through R9.
      G 00000004 00000000
      G 00000005 00000000
      G 00000006 00000000
      G 00000007 00000000
      G 00000008 00000000
      G 00000009 801D9000

>>> EXAMINE PR$ SCBB                         ! Examine the SCBB, IPR 17
      I 00000011 2004A000                    ! (decimal).

>>> EXAMINE/P 0                             ! Examine local memory 0.
      P 00000000 00000000

>>> EXAMINE/INS 20040000                     ! Examine 1st instruction
                                              ! in ROM.

      P 20040000    11 BRB    20040019

>>> EXAMINE/INS/N:5 20040019                 ! Disassemble from branch.
      P 20040019    D0 MOVL   I^#20140000,@#20140000
      P 20040024    D2 MCOML  @#20140030,@#20140502
      P 2004002F    D2 MCOML  S^#0E,@#20140030
      P 20040036    7D MOVQ   R0,@#201404B2
      P 2004003D    D0 MOVL   I^#201404B2,R1
      P 20040044    DB MFPR   S^#2A,B^44(R1)

>>> EXAMINE/INS                               ! Look at next instruction.
      P 20040048    DB MFPR   S^#2B,B^48(R1)
>>>
```

### 3.9.6 EXIT

The **EXIT** command exits from maintenance mode (>>>) to the normal mode prompt (>>). This command idles the CVAX chip. The **maint** command, typed in lowercase letters from the normal mode prompt, returns you to maintenance mode.

*Format:*

**EXIT**

*Example:*

```
>>> EXIT          ! From maintenance mode, exit to
                  ! normal mode.
>> maint          ! From normal mode, exit to
                  ! maintenance mode.
>>>
```

### 3.9.7 FIND

The **FIND** command searches main memory starting at address zero for a page-aligned 128-Kbyte segment of good memory, or a restart parameter block (RPB). If the command finds the segment or RPB, its address plus 512 is left in SP (R14). If it does not find the segment or RPB, the console issues an error message and preserves the contents of SP. If you do not specify a qualifier, /RPB is assumed.

*Format:*

**FIND [qualifier-list]**

*Qualifiers:*

*Command specific:*

- /MEMORY** Searches memory for a page-aligned block of good memory, 128 Kbytes in length. The search looks only at memory that is deemed usable by the bitmap. This command leaves the contents of memory unchanged.
- /RPB** Searches all of physical memory for an RPB. The search does not use the bitmap to qualify which pages are looked at. The command leaves the contents of memory unchanged.

*Examples:*

>>> EXAMINE SP	! Check the SP.
G 0000000E 00000000	
>>> FIND /MEM	! Look for a valid 128 Kbyte.
>>> EXAMINE SP	! Note where it was found.
G 0000000E 00000200	
>>> FIND /RPB	! Check for valid RPB.
?2C FND ERR 0CC00004	! None to be found here.
>>>	

### 3.9.8 HALT

The **HALT** command has no effect. It is included for compatibility with other VAX consoles.

*Format:*

**HALT**

*Example:*

```
>>> HALT                ! Pretend to halt.  
>>>
```

### 3.9.9 HELP

The HELP command provides information about command syntax and usage.

*Format:*

**HELP**

*Example:*

>>> HELP

Following is a brief summary of all the commands supported by the console:

UPPERCASE	denotes a keyword that you must type in
	denotes an OR condition
[]	denotes optional parameters
< >	denotes a field that must be filled in with a syntactically correct value

Valid qualifiers:

/B /W /L /Q /INSTRUCTION  
/G /I /V /P /M  
/STEP: /N: /NOT  
/WRONG /U

Valid commands:

DEPOSIT [qualifiers] <address> [datum [datum]]  
EXAMINE [qualifiers] [address]  
MOVE [qualifiers] <address>  
<address>  
SEARCH [qualifiers] <address>  
<pattern> [mask]  
SET BFLAG <boot\_flags>  
SET BOOT <boot\_device>  
SET HOST/DUP/DSSI <node\_number> [task]  
SET HOST/DUP/UQSSP </DISK /TAPE>  
<controller\_number> [task]  
SET HOST/DUP/UQSSP <physical\_CSR\_address> [task]  
SET HOST/MAINTENANCE/UQSSP/SERVICE <controller\_number> [task]  
SET HOST/MAINTENANCE/UQSSP <physical\_CSR\_address> [task]  
SET LANGUAGE <language\_number>



SHOW BFLAG  
SHOW BOOT  
SHOW DEVICE  
SHOW DSSI  
SHOW ETHERNET  
SHOW LANGUAGE  
SHOW MEMORY [/FULL]  
SHOW QBUS  
SHOW RLV12  
SHOW UQSSP  
SHOW VERSION  
HALT  
INITIALIZE  
UNJAM  
CONTINUE  
START <address>  
REPEAT <command>  
X <address> <count>  
FIND [/MEMORY or /RPB]  
TEST [test\_code [parameters]]  
BOOT [/R5:<boot\_flags> or /<boot\_flags>] [boot\_device]  
NEXT [count]  
EXIT  
CONFIGURE  
HELP  
>>>

### 3.9.10 INITIALIZE

The INITIALIZE command performs a processor initialization.

*Format:*

#### INITIALIZE

The following registers are initialized:

Register	State at Initialization
PSL	041F0000
IPL	1F
ASTLVL	4
SISR	0
ICCS	Bits <6> and <0> clear; the rest are unpredictable
RXCS	0
TXCS	80
MAPEI <sub>n</sub>	0
CVAX cache	Disabled, all entries invalid
Instruction buffer	Unaffected
Console previous reference	Longword, physical, address 0
TODR	Unaffected
Main memory	Unaffected
General registers	Unaffected
Halt code	Unaffected
Bootstrap-in-progress flag	Unaffected
Internal restart-in-progress flag	Unaffected

The firmware clears all error status bits and initializes as follows:

1. Clears any interrupts.
2. Initializes `pr$_scbb` to console SCB.
3. Initializes the IPR:

```
ipri$1_ipr           = 0
ipri$1_val           = 4
ipri$0_ipri          = 8
pr$_ipl              = ^x0000001F
pr$_actlvi           = ^x00000004
pr$_eier             = ^x00000030
pr$_iccs             = ^x00000000
pr$_racs             = ^x00000000
pr$_txcs             = ^x00000080
pr$_mapen            = ^x00000000
ctx_base plus ctx$1_pal = ^x041F0000
```

4. Flushes and disables the chip cache.
5. Initializes the SSC.
6. Initializes the console state:

Sets the current and previous reference to **PHYSICAL** and **LONG** at address 0; current datum is then 0; clears the exit flag:

```
ca_base plus ca$1_address      = 0
ca_base plus ca$1_prev_address = 0
ca_base plus ca$1_datum_size   = 4
ca_base plus ca$q_datum        = 0
ca_base plus ca$1_qv           = 0
plus 4*qual$u_n                = 0
ctx_base plus ctx$b_exit       = 0
```

*Example:*

```
>>> INIT
>>>
```

### 3.9.11 MOVE

The MOVE command copies the block of memory starting at the source address to a block beginning at the destination address. Typically, this command has an /N qualifier so that more than one datum is transferred. The destination correctly reflects the original contents of the source, regardless of the overlap between the source and the data.

The MOVE command actually performs byte, word, longword, and quadword reads and writes as needed in the process of moving the data. Moves are supported only for the physical and virtual address spaces.

*Format:*

**MOVE [qualifier-list] (src\_address) (dest\_address)**

*Qualifiers:*

*Data control:* /B, /W, /L, /Q, /N:(count), /STEP:(size), /WRONG

*Address space control:* /V, /U, /P

*Arguments:*

- |                |  |
|----------------|--|
| (src_address)  | A longword address that specifies the first location of the source data to be copied.  |
| (dest_address) | A longword address that specifies the destination of the first byte of data. These addresses may be an actual address or a symbolic address. If no address is specified, + is assumed. |

### **Examples:**

```
>>> EXAMINE/N:4 0                                ! Observe destination.
P 00000000 00000000
P 00000004 00000000
P 00000008 00000000
P 0000000C 00000000
P 00000010 00000000

>>> EXAMINE/N:4 200                              ! Observe source data.
P 00000200 58DD0520
P 00000204 585E04C1
P 00000208 00FF8FBB
P 0000020C 5208A8D0
P 00000210 540CA8DE

>>> MOVE/N:4 200 0                               ! Move the data.

>>> EXAMINE/N:4 0                                ! Observe moved data.
P 00000000 58DD0520
P 00000004 585E04C1
P 00000008 00FF8FBB
P 0000000C 5208A8D0
P 00000010 540CA8DE
>>>
```

### 3.9.12 NEXT

The NEXT command executes the specified number of macro instructions. If no count is specified, 1 (one) is assumed.

After the last macro instruction is executed, the console reenters maintenance mode.

*Format:*

**NEXT [count]**

The console implements the NEXT command, using the trace trap enable and trace pending bits in the PSL and the trace pending vector in the SCB. The following restrictions apply:

- If memory management is enabled, the NEXT command works only if the first page in SSC RAM is mapped in S0 (system) space.
- Overhead associated with the NEXT command affects execution time of an instruction.
- The NEXT command elevates the IPL to 31 for long periods of time (milliseconds) while single-stepping over several commands.
- Unpredictable results occur if the macro instruction being stepped over modifies either the SCBB or the trace trap entry. This means that you cannot use the NEXT command in conjunction with other debuggers.

*Arguments:*

[count]      A value representing the number of macro instructions to execute.

*Examples:*

```
>>> EXAMINE PC
      G 0000000F 00000200
>>> NEXT
      PC = 00000202
>>> NEXT 4
      PC = 00000213
>>>
```

```

>>> EXAMINE/INS/N:10 0
P 00000000 01 NOP
P 00000001 01 NOP
P 00000002 01 NOP
P 00000003 01 NOP
P 00000004 01 NOP
P 00000005 01 NOP
P 00000006 01 NOP
P 00000007 01 NOP
P 00000008 11 BRB 00000002
P 0000000A 01 NOP
P 0000000B 01 NOP
P 0000000C 00 HALT
P 0000000D 00 HALT
P 0000000E 00 HALT
P 0000000F 00 HALT
P 00000010 00 HALT
P 00000011 00 HALT

```

```

>>> DEP PC 0

```

```

>>> N
P 00000001 01 NOP
>>> N
P 00000002 01 NOP
>>> N
P 00000003 01 NOP
>>> N
P 00000004 01 NOP
>>> N
P 00000005 01 NOP
>>> N 5
P 00000006 01 NOP
P 00000007 01 NOP
P 00000008 11 BRB 00000002
P 00000002 01 NOP
P 00000003 01 NOP

```

### 3.9.13 REPEAT

The REPEAT command repeatedly displays and executes the specified command. Press **CWC** to stop the command. You can specify any valid console command except the REPEAT command.

*Format:*

**REPEAT (command)**

*Arguments:*

(command) A valid console command other than REPEAT.

*Examples:*

```
>>> REPEAT EXAMINE PRS_TODR          ! Watch the clock.
I 0000001B 5AFE78CE
I 0000001B 5AFE78D1
I 0000001B 5AFE78FD
I 0000001B 5AFE7900
I 0000001B 5AFE7903
I 0000001B 5AFE7907
I 0000001B 5AFE790A
I 0000001B 5AFE790D
I 0000001B 5AFE7910
I 0000001B 5AFE793C
I 0000001B 5AFE793F
I 0000001B 5AFE7942
I 0000001B 5AFE7946
I 0000001B 5AFE7949
I 0000001B 5AFE794C
I 0000001B 5AFE794F
I 0000001B 5^C
>>>
```



### 3.9.14 SEARCH

The **SEARCH** command finds all occurrences of a pattern and reports the addresses where the pattern was found. If the **/NOT** qualifier is present, the command reports all addresses in which the pattern did not match.

*Format:*

**SEARCH** [qualifier\_list] (address) (pattern) [(mask)]

**SEARCH** accepts an optional mask that indicates bits to be ignored (*don't care* bits). For example, to ignore bit 0 in the comparison, specify a mask of 1. The mask, if not present, defaults to 0.

A match occurs if (pattern AND mask complement) = (data AND mask complement), where:

pattern is the target data

mask is the optional don't care bitmask (which defaults to 0)

data is the data at the current address

**SEARCH** reports the address under the following conditions:

<b>/NOT Qualifier</b>	<b>Match Condition</b>	<b>Action</b>
Absent	True	Report address
Absent	False	No report
Present	True	No report
Present	False	Report address

The address is advanced by the size of the pattern (byte, word, longword, or quadword), unless overridden by the **/STEP** qualifier.

*Qualifiers:*

*Data control:* **/B**, **/W**, **/L**, **/Q**, **/N:(count)**, **/STEP:(size)**, **/WRONG**

*Address space control:* **/P**, **/N**, **/U**

*Command-specific:*

**/NOT**            Inverts the sense of the match.

## Arguments:

(start\_address) A longword address that specifies the first location subject to the search. This address can be an actual address or a symbolic address. If no address is specified, + is assumed.

(pattern) The target data.

((mask)) A longword containing the bits desired in the comparison.

## Examples:

```
>>> DEPOSIT /P/L/N:1000 0 0          !Clear some memory.
>>>
>>> DEPOSIT 300 12345678             !Deposit some search data.
>>> DEPOSIT 401 12345678
>>> DEPOSIT 502 87654321
>>>

>>> SEARCH /N:1000 /ST:1 0 12345678 !Search for all occurrences
    P 00000300 12345678             !of 12345678 on any byte
    P 00000401 12345678             !boundary.

>>> SEARCH /N:1000 0 12345678        !Try on longword boundaries.
    P 00000300 12345678

>>> SEARCH /N:1000 /NOT 0 0          !Search for all non-zero
    P 00000300 12345678             !longwords.
    P 00000400 34567800
    P 00000404 00000012
    P 00000500 43210000
    P 00000504 00008765

>>> SEARCH /N:1000 /ST:1 0 1 FFFFFFFE !Search for odd longwords
    P 00000502 87654321             !on any boundary.
    P 00000503 00875543
    P 00000504 00008765
    P 00000505 00000087

>>> SEARCH /N:1000 /B 0 12           !Search for all occurrences
    P 00000303 12                   !of the byte 12.
    P 00000404 12

>>> SEARCH /N:1000 /ST:1 /W 0 FE11  !Search for all words that
>>>                                  !could be interpreted as a
>>>                                  !spin (10$:brb 10$).
>>>                                  !None were found.
```

### 3.9.15 SET

The SET command sets the parameter to the value you specify.

*Format:*

**SET (parameter) (value)**

*Parameters:*

- BFLAG** Sets the default R5 boot flags. The value must be a hex number of up to 8 digits. See Table 3-4, VMB Boot Flags, for a list of the boot flags.
- BOOT** Sets the default boot device. The value must be a valid device name as specified in the BOOT command description, Section 3.9.1.
- HOST** Connects to the DUP or MAINTENANCE driver on the selected node or device. Note the hierarchy of the SET HOST qualifiers below.
- /DUP**—Uses the DUP driver to execute local programs of a device on either the DSSI bus or the Q22-bus.
- /DSSI node**—Attaches to the DSSI node. A node is a name up to 8 characters in length or a number from 0 to 7.
- /UQSSP**—Attaches to the UQSSP device specified, using one of the following methods:
- /DISK n**—Specifies the disk controller number, where n is a number from 0 to 255. The resulting fixed address for n=0 is 20001468 for the first TMSCP controller, and the floating rank for n>0 is 26.
  - /TAPE n**—Specifies the tape controller number, where n is a number from 0 to 255. The resulting fixed address for n=0 is 20001940 for the first MSCP controller, and the floating rank for n>0 is 30.
  - /csr\_address**—Specifies the Q22-bus I/O page CSR address for the device.
- /MAINTENANCE**—Examines and modifies DSSI controller module configuration values. Does not accept a task value.
- /UQSSP—**
- /SERVICE n**—Specifies service for DSSI controller module n, where n is a value from 0 to 3. (The resulting fixed address of a DSSI controller module in maintenance mode is 20001910+4\*n.)
  - /csr\_address**—Specifies the Q22-bus I/O page CSR address for the DSSI controller module.
- LANGUAGE** Sets console language and keyboard type. If the current console terminal does not support the Digital Multinational Character Set (MCS), then this command has no effect and the console message appears in English. Values are 1 through 15. Refer to Example 3-1 for the languages you can select.

*Qualifiers:* Listed in the parameter descriptions above.

### Examples:

```
>>> SET BFLAG 220      ! Sets boot flags 5 and 9 (See boot flag
                        ! table in the BOOT command description.)
>>> SET BOOT DUA0
```

```
>>> SET HOST/DUP/DSSI 0
Starting DUP server...
```

DSSI Node 0 (SUSAN)

DRVEXR V1.0 D 2-JUN-1989 10:01:35

DRVSTST V1.0 D 2-JUN-1989 10:01:35

HISTORY V1.0 D 2-JUN-1989 10:01:35

ERASE V1.0 D 2-JUN-1989 10:01:35

PARAMS V1.0 D 2-JUN-1989 10:01:35

DIRECT V1.0 D 2-JUN-1989 10:01:35

Copyright © 1988 Digital Equipment Corporation

Task Name? PARAMS

Copyright © 1988 Digital Equipment Corporation

PARAMS> STAT PATH

ID	Path	Block	Remote	Node	DGS_S	DGS_R	MSG_S	MSG_R
---	---	---	---	---	---	---	---	---
0	PB	FF811ECC	Internal	Path	0	0	0	0
1	PB	FF8120D4	KAREN	RFX V101	0	0	0	0
4	PB	FF8121D8	WILMA	RFX V101	0	0	0	0
5	PB	FF8120DC	BETTY	RFX V101	0	0	0	0
2	PB	FF8122E0	DSSI1	VMS V5.0	0	0	816	3045
3	PB	FF8124E4	3	VMB BOOT	0	0	50	52

PARAMS> EXIT

Exiting...

Task Name?

Stopping DUP server...

>>> SET HOST/DUP/DSSI 0 PARAMS

Starting DUP server...

DSSI Node 0 (SUSAN)

Copyright © 1988 Digital Equipment Corporation

PARAMS> SHOW NODE

Parameter	Current	Default	Type	Radix
NODENAME	SUSAN	RF30	String	Ascii B

PARAMS> SHOW ALLCLASS

Parameter	Current	Default	Type	Radix
ALLCLASS	1	0	Byte	Dec B

PARAMS> EXIT

Exiting...

Stopping DUP server...

>>>

### 3.9.16 SHOW

The SHOW command displays the console parameter you specify.

*Format:*

**SHOW** (parameter)

*Parameters:*

**BFLAG** Displays the default R5 boot flags.

**BOOT** Displays the default boot device.

**DEVICE** Displays all devices displayed by the SHOW DSSI, SHOW ETHERNET, and SHOW UQSSP commands.

**DSSI** Displays the status of all nodes that can be found on the DSSI bus. For each node on the DSSI bus, the firmware displays the node number, the node name, and the boot name and type of the device, if available. The command does not indicate whether the device contains a bootable image.

The node that issues the command is listed with a node name of \* (asterisk).

The device information is obtained from the media type field of the MSCP command GET UNIT STATUS. If a node is not running or is not capable of running an MSCP server, then no device information is displayed.

**ETHERNET** Displays hardware Ethernet address for all Ethernet adapters that can be found, both on-board and on the Q22-bus. Displays as blank if no Ethernet adapter is present.

**LANGUAGE** Displays console language and keyboard type. Refer to the corresponding SET LANGUAGE command for the meaning.

**MEMORY** Displays main memory configuration board-by-board.

**/FULL**—Additionally, displays the normally inaccessible areas of memory, such as the PFN bitmap pages, the console scratch memory pages, the Q22-bus scatter-gather map pages. Also reports the addresses of bad pages, as defined by the bitmap.

**QBUS** Displays all Q22-bus I/O addresses that respond to an aligned word read, and vector and device name information. For each address, the console displays the address in the VAX I/O space in hex, the address as it would appear in the Q22-bus I/O space in octal, and the word data that was read in hex. Also displays the vector that you should set up and device name(s) that could be associated with the CSR.

This command may take several minutes to complete. Press **Ctrl/C** to terminate the command. During execution, the command disables the scatter-gather map so that it can search for memory on the Q-bus.

**RLV12**      Displays all RL01 and RL02 disks that appear on the Q22-bus.

**UQSSP**      Displays the status of all disks and tapes that can be found on the Q22-bus that support the UQSSP protocol. For each such disk or tape on the Q22-bus, the firmware displays the controller number, the controller CSR address, and the boot name and type of each device connected to the controller. The command does not indicate whether the device contains a bootable image.

This information is obtained from the media type field of the MSCP command GET UNIT STATUS. The console does not display device information if a node is not running (or cannot run) an MSCP server.

**VERSION**    Displays the current firmware version.

**Qualifiers:** Listed in the parameter descriptions above.

**Examples:**

```
>>> SHOW BFLAG
00000220
```

```
>>> SHOW BOOT
DUA0
```

```
>>> SHOW DEVICE
>>>show device
DSSI Node 0 (SUSAN)
-DIA0 -rf(0,0,*) (RF71)
```

```
DSSI Node 1 (KAREN)
-DIA1 -rf(1,1,*) (RF71)
```

```
DSSI Node 7 (*)
```

```
UQSSP Tape Controller 0 (772150)
-MUA0 -tm(0,0) (TK70)
```

```
Ethernet Adapter
-ESA0 -se -mop() (08-00-2B-0C-C4-75)
```

```
>>> SHOW DSSI
DSSI Node 0 (SUSAN)
-DIA0 -rf(0,0,*) (RF71)

DSSI Node 1 (KAREN)
-DIA1 -rf(1,1,*) (RF71)

DSSI Node 7 (*)

>>> SHOW ETHERNET
Ethernet Adapter
-ESA0 -se -mop() (08-00-2B-0C-C4-75)

>>> SHOW LANGUAGE
English (United States/Canada)

>>> SHOW MEMORY
Memory 0: 00000000 to 003FFFFFF, 4MB, 0 bad pages
Total of 4MB, 0 bad pages, 98 reserved pages

>>> SHOW MEMORY/FULL
Memory 0: 00000000 to 003FFFFFF, 4MB, 0 bad pages
Total of 4MB, 0 bad pages, 98 reserved pages

Memory Bitmap
-003F8000 to 003FFFFFF, 2 pages

Console Scratch Area
-003F4000 to 003F7FFF, 32 pages

Qbus Map
-003F8000 to 003FFFFFF, 64 pages

Scan of Bad Pages
```



>>> SHOW QBUS

Scan of Qbus I/O Space

-20001468 (772150) = 4000 (154) RQDX3/KDA50/RRD50/RQC25/X-DISK  
-2000146A (772152) = 0B40  
-20001940 (774500) = 0000 (260) TQK50/TQK70/TU81E/RV20/X-TAPE  
-20001942 (774502) = 0BC0  
-20001F40 (777500) = 0020 (004) IPCR

Scan of Qbus Memory Space

>>> SHOW UQSSP

UQSSP Tape Controller 0 (772150)

-MUA0 -tm(0,0) (TK70)

>>> SHOW VERSION

KN210--A Vn.n

>>>

### 3.9.17 START

The **START** command starts instruction execution at the address you specify. If no address is given, the current PC is used. If memory mapping is enabled, macro instructions are executed from virtual memory, and the address is treated as a virtual address. The **START** command is equivalent to a **DEPOSIT** to PC, followed by a **CONTINUE**. The **START** command does not perform a processor initialization.

*Format:*

**START** [address]

*Arguments:*

[address]      The address at which to begin execution. This address is loaded into the user's PC.

*Example:*

```
>>> START 1000
```

### 3.9.18 TEST

The **TEST** command invokes a diagnostic test program specified by the test number. If you enter a test number of 0 (zero), all tests allowed to be executed from the console terminal are executed. The console accepts an optional list of up to five additional hexadecimal arguments.

Refer to Chapter 4 for a detailed explanation of the diagnostics.

#### *Format:*

**TEST (test\_number) [test\_arguments]**

#### *Arguments:*

- |                         |   |
|-------------------------|---|
| <b>(test_number)</b>    | A two-digit hex number specifying the test to be executed.  |
| <b>(test_arguments)</b> | Up to five additional test arguments. These arguments are accepted but they have no meaning to the console. |

#### *Example:*

```
>>> TEST 0                !Execute the power-up diagnostic script.
                           !Test 0 has the same effect as a power-up.
51..50..49..48..47..46..45..44..43..42..41..40..39..38..37..36..35..
34..33..32..31..30..29..28..27..26..25..24..23..22..21..20..19..18..
17..16..15..14..13..12..11..10..09..08..07..06..05..04..03..
>>>
```

### **3.9.19 UNJAM**

The UNJAM command performs an I/O bus reset, by writing a 1 (one) to IPR 55 (decimal).

*Format:*

**UNJAM**

*Example:*

```
>>> UNJAM  
>>>
```

### 3.9.20 X—Binary Load and Unload

The X command is for use by automatic systems communicating with the console.

The X command loads or unloads (that is, writes to memory or reads from memory) the specified number of data bytes through the console serial line (regardless of console type), starting at the specified address.

*Format:*

**X (address) (count) CR (line\_checksum) (data) (data\_checksum)**

If bit 31 of the count is clear, data is received by the console and deposited into memory. If bit 31 is set, data is read from memory and sent by the console. The remaining bits in the count are a positive number indicating the number of bytes to load or unload.

The console accepts the command upon receiving the carriage return. The next byte the console receives is the command checksum, which is not echoed. The command checksum is verified by adding all command characters, including the checksum and separating space (but not including the terminating carriage return, rubouts, or characters deleted by rubout), into an 8-bit register initially set to zero. If no errors occur, the result is zero. If the command checksum is correct, the console responds with the input prompt and either sends data to the requester or prepares to receive data. If the command checksum is in error, the console responds with an error message. The intent is to prevent inadvertent operator entry into a mode where the console is accepting characters from the keyboard as data, with no escape mechanism possible.

If the command is a load (bit 31 of the count is clear), the console responds with the input prompt (>>>), then accepts the specified number of bytes of data for depositing to memory, and an additional byte of received data checksum. The data is verified by adding all data characters and the checksum character into an 8-bit register initially set to zero. If the final content of the register is nonzero, the data or checksum is in error, and the console responds with an error message.

If the command is a binary unload (bit 31 of the count is set), the console responds with the input prompt (>>>), followed by the specified number of bytes of binary data. As each byte is sent, it is added to a checksum register initially set to zero. At the end of the transmission, the two's complement of the low byte of the register is sent.

If the data checksum is incorrect on a load or if memory or line errors occur during the transmission of data, the entire transmission is completed, then the console issues an error message. If an error occurs during loading, the contents of the memory being loaded are unpredictable.

The console represses echo while it is receiving the data string and checksums.

The console terminates all flow control when it receives the carriage return at the end of the command line in order to avoid treating flow control characters from the terminal as valid command line checksums.

You can control the console serial line during a binary unload, using control characters (Ctrl/C, Ctrl/S, Ctrl/O, and so on). You cannot control the console serial line during a binary load, since all received characters are valid binary data.

The console has the following timing requirements:

- It must receive data being loaded with a binary load command at a rate of at least one byte every 60 seconds.
- It must receive the command checksum that precedes the data within 60 seconds of the carriage return that terminates the command line.
- It must receive the data checksum within 60 seconds of the last data byte.

If any of these timing requirements is not met, the console aborts the transmission by issuing an error message and returning to the console prompt.

The entire command, including the checksum, can be sent to the console as a single burst of characters at the specified character rate of the console serial line. The console is able to receive at least 4 Kbytes of data in a single X command.

### **3.9.21 ! (Comment)**

The comment character (an exclamation point) is used to document command sequences. It can appear anywhere on the command line. All characters following the comment character are ignored.

*Format:* !

*Example:*

```
>>> ! The console ignores this line.  
>>>
```





## Chapter 4

# Troubleshooting and Diagnostics

---

### 4.1 Introduction

This chapter contains a description of KN210 ROM-based diagnostics, acceptance test and troubleshooting procedures, diagnostics, and power-up self-tests for common options.

### 4.2 General Procedures

Before troubleshooting any system problem, check the site maintenance guide for the system's service history. Ask the system manager two questions:

- Has the system been used before and did it work correctly?
- Have changes been made to the system recently?

Three problems commonly occur when you make a change to the system:

- Incorrect cabling
- Module configuration errors (incorrect CSR addresses and interrupt vectors)
- Incorrect grant continuity

Most communications modules use floating CSR addresses and interrupt vectors. If you remove a module from the system, you may have to change the addresses and vectors of other modules. *Microsystems Options* lists address and vector values for most options.

If you change the system configuration, run the CONFIGURE utility at the maintenance mode prompt (>>>) to determine the CSR addresses and interrupt vectors recommended by Digital.

See the *MicroVAX Diagnostic Monitor User's Guide* for information about the CONNECT and IGNORE commands, which are used to set up MDM for testing nonstandard configurations.

See Appendix A for a summary of ULTRIX-32 Exerciser and Uerf commands to help you troubleshoot and diagnose errors. When

troubleshooting, note the status of cables and connectors before you perform each step. Label cables before you disconnect them to save time and prevent you from introducing new problems.

If the system fails (or appears to fail) to boot the operating system, check the console terminal screen for an error message. If the terminal displays an error message, see Section 4.3. Check the LEDs on the device you suspect is bad. If no errors are indicated by the device LEDs, run the ROM-based diagnostics described in this chapter. In addition, check the following:

- If the system DC OK LED remains off, check the power supply and power supply cabling.
- If no message appears, make sure the console terminal and the system are on. Check the on/off power switch on both the console terminal and the system. If the terminal has a DC OK LED, be sure it is on.
- Check the cabling to the console terminal.
- Check the hex display on the H3602-AB. If the display is off, check the CPU module LEDs and the CPU cabling. If a hex error message appears on the H3602-AB or the module, see Section 4.3. On power-up, the display changes to reflect the power-up self-tests (see Section 4.3.5).
- If you cannot get a display of any kind on the console terminal, try another terminal.

If the system boots successfully, but a device seems to fail or an intermittent failure occurs, check the error log first for a device problem. The failing device is usually in one of the following areas:

- CPU
- Memory
- Mass storage
- Communications devices

## 4.3 KN210 CVAX ROM-Based Diagnostics

The KN210 CVAX ROM-based diagnostic facility, rather than the MicroVAX Diagnostic Monitor (MDM), is the primary diagnostic tool for troubleshooting and testing of the CPU, memory, Ethernet, and DSSI subsystems. ROM-based diagnostics have significant advantages:

- Load time is virtually nonexistent.
- The boot path is more reliable.
- Diagnosis is done in a more primitive state. (MDM requires successful loading of the operating system )

The ROM-based diagnostics can indicate several different FRUs, not just the CPU module. For example, they can isolate one of up to four memory modules as FRUs. (Table 4-7 lists the FRUs indicated by ROM-based diagnostic error messages.)

The diagnostics run automatically on power-up. While the diagnostics are running, the LEDs on the H3602-AB display a hexadecimal countdown of the tests from F to 4 (though not in precise reverse order) before booting the operating system, and 2 to 0 while booting the operating system. A different countdown appears on the console terminal.

The ROM-based diagnostics are a collection of individual tests with parameters that you can specify. A data structure called a *script* points to the tests. (See Section 4.3.2.) There are several field and manufacturing scripts. Qualified Customer Services personnel can also create their own scripts interactively.

A program called the *diagnostic executive* determines which of the available scripts to invoke. The script sequence varies if the KN210 is in a manufacturing environment. The diagnostic executive interprets the script to determine what tests to run, the correct order to run the tests, and the correct parameters to use for each test.

The diagnostic executive also controls tests so that errors can be detected and reported. It also ensures that when the tests are run, the machine is left in a consistent and well-defined state.

### 4.3.1 Diagnostic Tests

Table 4–1 shows a list of the CVAX ROM-based tests and utilities. To get this listing, enter `T 92` at the maintenance mode prompt (T is the abbreviation of TEST). The column headings have the following meanings:

- **Test** is the test code or utility code.
- **Address** is the test or utility's base address in ROM. This address varies. The addresses shown are only examples. If a test fails, entering `T FE` displays the diagnostic state to the console. You can subtract the base address of the failing test from the `last_exception_pc` to find the index into the failing test's diagnostic listing.
- **Name** is a brief description of the test or utility.
- **Parameters** shows the parameters for each diagnostic test or utility. Tests accept up to 10 parameters. The asterisks (\*) represent parameters that are used by the tests but that you cannot specify individually. These parameters are encoded in ROM and are provided by the diagnostic executive.

**Table 4-1: Test and Utility Numbers**

Test	Address	Name	Parameters
	2004C000	De_SCB	
	2004B200	CP_SCB	
C6	2004D6DC	SSC_powerup	*****
C7	2004D79E	CBTCR timeout	***
34	2004D858	ROM logic test	*
33	2004D920	CMCTL_powerup	*
32	2004D968	CMCTL regs	MEMCSR0_addr *****
91	2004DA8C	CQBIC_powerup	**
90	2004DB1C	CQBIC regs	*
80	2004DB9E	CQBIC_memory	*****
60	2004E1C5	Console serial	start_baud end_baud *****
52	2004E512	Prog timer	which_timer wait_time_us ***
53	2004F7D8	TOY clock	repeat_test_250ms ea tolerance ***
55	2004E983	Interval timer	*
5A	2004E9F8	VAX CMCTL CDAL	don't_report_memory_bad repeat_count *
45	2004EB0A	Cache_mem_cqbic	start_addr end_addr addr_incr ****
46	2004FDF0	Cache1_diag_md	addr_incr *****
9E	2004F41A	List diag	*
81	2004F440	MSCP-QBUS test	IP_csr *****
82	2004F602	DELQA	device_num_addr ****
C1	2004F7DD	SSC RAM	*
C2	2004F9A4	SSC RAM ALL	*
C5	2004FB20	SSC regs	*
56	2004FC14	SII_ext_loopbck	***
5C	2004FF19	SII_initiator	run_test *****
5D	20040C0C	SII target	run_test *****
58	200523D2	DSSI reset	port_no time_secs *
57	200527B8	SII_memory	incr test_pattern run_test *****
5E	200527CD	NI_memory	incr test_pattern run_test *****
5B	20052BF6	SII_registers	run_test ****
5F	20052DC4	NI_test	do_extl where run_test *****
54	2005392E	Virtual mode	*****
41	20053C04	Board reset	***
42	20053DB1	Check_for_intrs	***
31	20053D4F	MEM_setup_CSRs	*****
30	20054323	MEM_bitmap	*** mark_hard_SBEs *****
4F	2005475E	MEM_data	start_addr end_addr addr_incr cont_on_err *****
4E	2005481C	MEM_byte	start_addr end_addr addr_incr cont_on_err *****
4D	20054A31	MEM_address	start_addr end_addr addr_incr cont_on_err *****
4C	20054BC3	MEM_ECC_error	start_addr end_addr addr_incr cont_on_err *****

**Table 4–1 (Cont.): Test and Utility Numbers**

<b>Test</b>	<b>Address</b>	<b>Name</b>	<b>Parameters</b>
4B	20055084	MEM_maskd_errs	start_add end_add add_incr cont_on_err *****
4A	2005525E	MEM_correction	start_add end_add add_incr cont_on_err *****
49	20055471	MEM_FDM_logic	*** cont_on_err *****
48	20055A40	MEM_addr_shrts	start_add end_add * cont_on_err pat2 pat3 ****
47	20056097	MEM_refresh	start end incr cont_on_err time_seconds *****
40	20056222	MEM_count_errs	First_board Last_board Soft_errs_allowed *****
44	2005653C	Cache_memory	addr_incr *****
9D	20056888	Utilities	Expnd_err_mag get_mode init_LEDs clr_ps_ cnt
9C	2005698C	List CPU regs	*
9F	200572AF	Create script	*****
50	20057A0F	M7636 present	*
70	20057A3C	R3000_cache	***
71	20057A53	R3000_fpu	*
72	20057A68	R3000_tlb	*
73	20057A7D	R3000_reg_intrf	loop run_test ***
74	20057A97	R3000_buf_intrf	increment pattern start_add end_add ***
75	20057AB1	R3000_mem_intrf	increment pattern start_add end_add ***
76	20057ACB	R3000_interrupt	run_test
77	20057AE5	R3000_mov_inver	increment * start_add end_add ***
78	20057AFF	R3000_write_buf	*

Parameters that you can specify are written out, as shown in the following examples:

```
54 2004E557 Virtual mode *****
30 20053C6D MEM_bitmap *** mark_Hard_SBEs *****
```

The virtual mode test on the first line contains several parameters, but you cannot specify any of them. To run this test individually, enter:

```
>>> T 54
```

The MEM\_bitmap test on the second line accepts ten parameters, but you can specify only the fourth one. To mark pages bad in the bitmap for single-bit or multibit errors, enter a 1 in the fourth parameter field:

```
>>> T 30 0 0 0 1
```

You must enter a value of either 0 (zero) or 1 (one) for the first three parameters. (0 is used in this example.) The values have no effect on the test; they are simply placeholders for the first three parameters. You

do not have to specify a value for parameters that follow the user-defined parameter.

### 4.3.2 Scripts

Most of the tests shown by utility 9E are arranged into scripts. A *script* is a data structure that points to various tests and defines the order in which they are run. Different scripts can run the same set of tests, but in a different order and/or with different parameters and flags. A script also contains the following information:

- The parameters and flags that need to be passed to the test.
- Where the tests can be run from. For example, certain tests can be run only from the EPROM. Other tests are program-independent code, and can be run from EPROM, cache diagnostic space, or main memory to enhance execution speed.
- What is to be shown, if anything, on the console.
- What is to be shown, if anything, in the LED display.
- What action to take on errors (halt, repeat, continue).

The power-up script runs every time the system is powered on. You can also invoke the power-up script at any time by entering T 0.

Additional scripts are included in the ROMs for use in manufacturing and engineering environments.

Customer Services personnel can run scripts and tests individually, using the T command. When doing so, note that certain tests may be dependent upon a state set up from a previous test. For this reason you should use the UNJAM and INITIALIZE commands, described in Chapter 3, before running an individual test after the operating system has crashed or has been halted. You do not need to use these commands on system power-up, however, because system power-up leaves the machine in a defined state.

Customer Services personnel with a detailed knowledge of the 210 hardware and firmware can also create their own scripts by using the 9F utility. (See Section 4.3.4.)

Table 4-2 lists the scripts that are available to Customer Services.

**Table 4-2: Scripts Available to Customer Services**

Script <sup>1</sup>	Enter with TEST Command	Description
A0	A0	Soft script created by de_test9f. Enter T 9F to create.
A1	A1, AA, AB, AC, 0, 3	Common section of power-up script. Scripts AA, AB, and AC invoke this script at power-up.
A7	A7, A8	Memory test portion invoked by Script A8. Reruns the memory tests without rebuilding and reinitializing the bitmap. Run Script A8 once before running Script A7 separately to allow mapping out of both single-bit and double-bit main memory ECC errors.
A8	A8	Memory acceptance. Running Script A8 with Script A7 tests main memory more extensively. It enables hard single-bit and multibit main memory ECC errors to be marked bad in the bitmap. Invokes Script A7 when it has completed its tests.
A9	A9	Memory tests. Halts and reports the first error. Does not reset the bitmap or busmap.
AA	AA, 0	Console SLU. Invokes Scripts BA, BC, and A1. Does not invoke any tests directly.
AC	AC, 3	Power-up. Invokes Scripts BC and A1. Does not invoke any tests directly. Invoked at power-up.
AD	AD	Console program. Runs memory tests, marks bitmap, resets busmap, and resets caches. Calls Script AE.
AE	AE, AD	Console program. Resets memory CSRs and resets caches.
AF	AF	Console program. Resets busmap and resets caches.
BA	BA, 2, AA	Initial power-up script for console SLU before first console announcement. Invoked at power-up.
BC	BC, AA, AC, 0, 3	Called by Scripts AA and AC. Provides console announcements. Invoked at power-up.
BE	AA, A5, BD, BE	Runs R3000 CPU, floating-point unit, cache, TLB (translation lookaside buffer), and I/O module interface tests.
BF	AA BF	Runs KN210 I/O module tests.

<sup>1</sup>Scripts A2-A6, B0-B3, and B5 are for manufacturing use. They should not be used by Customer Services. Scripts AB and BB are used to test the QDSS, which is not supported. Scripts BD, B4, and B6-B9 are not used.



In most cases, Customer Services needs only the commands shown in Table 4-3 for effective troubleshooting and acceptance testing.

**Table 4-3: Commonly Used Field Service Scripts**

<b>Command</b>	<b>Description</b>
0	Automatically invokes the proper scripts; runs the same tests as during power-up.
A9	Primarily runs the memory tests; halts upon first hard or soft error.
A8	Memory acceptance script; marks hard multibit and single-bit ECC errors in the bitmap. Script A8 calls Script A7 when this command is entered. Script A7 contains the memory tests that will continue on error.
A7	Can be run by itself; useful when you want to bypass the bitmap test.
A1	Power-up script that can be run by itself. Bypasses the bitmap test.

### 4.3.3 Script Calling Sequence

#### Actions at Power-Up

In a nonmanufacturing environment where the intended console device is the serial line unit (SLU), the console program (referred to as CP below) performs the following actions at power-up:

1. Runs the IPT.
2. Assumes console device is SLU.
3. Calls the diagnostic executive (DE) with Test Code = 2.
  - a. DE determines that the environment is nonmanufacturing from H3602-AB. (Manufacturing sets a jumper on the H3602-AB for testing.)
  - b. DE selects script sequence for console SLU.
  - c. DE executes Script BA.
    - Script BA directs DE to execute test. (Console announcements are off.)
  - d. DE passes control back to the CP.
4. Establishes SLU as console device (whether or not SLU test passed).
5. Prints banner message.
6. Displays language inquiry menu on console if console supports MCS and any of the following is true:

- Battery is dead.
- H3602-AB switch set to action (language inquiry).
- Contents of SSC NVRAM are invalid.

**7. Calls DE with Test Code = 3.**

**a. DE executes Script AC.**

Script AC directs DE to execute Scripts BC and A1.

- Script BC directs DE to execute tests. (Console announcements are on.)
- Script A1 directs DE to execute tests. (Console announcements are on.)

**b. DE passes control back to CP.**

**8. CP issues end message and >>> prompt. CP may exit to >> prompt.**

**Actions After You Enter T 0**

In a nonmanufacturing environment where the intended console device is the SLU, the console program (CP) performs the following actions after you enter T 0 at the console prompt (>>> T 0):

**1. Calls the diagnostic executive (DE) with Test Code = 0.**

- a. DE determines environment is nonmanufacturing from H3602-AB switch setting.**
- b. DE executes Script AA.**

Script AA directs DE to execute Scripts BA, BC, and A1.

- Script BA directs DE to execute tests. (Console announcements are off.)
- Script BC directs DE to execute tests. (Console announcements are on.)
- Script A1 directs DE to execute tests. (Console announcements are on.)

**c. DE passes control back to the CP.**

**2. CP prints end message and >>> prompt. Console may exit to >> prompt.**

Note that although the sequence of actions is different in the two cases above, the same tests (those in Scripts BA, BC, and A1) are run both times.

### 4.3.4 Creating Scripts

You can create your own script, using utility 9F to control the order in which tests are run and to select specific parameters and flags for individual tests. In this way, you do not have to use the defaults provided by the hard-wired scripts.

Utility 9F also provides an easy way to see what flags and parameters are used by the diagnostic executive for each test.

Run test 9F first to build the user script (see Example 4-1). Press **[Return]** to use the default parameters or flags, which are shown in parentheses. Test 9F prompts you for the following information:

- **Script location.** The script can be located in the 1-Kbyte NVRAM in the SSC, in the 128-Kbyte mass storage interface (MSI) RAM in the SII chip, or in main memory. A script is limited by the size of the data structure that contains it. A larger script can be developed in main memory than in MSI RAM, and a larger script can be built in MSI RAM than in NVRAM.

A script cannot, however, always be located in main memory. For example, a script that runs memory tests will overwrite the user script, since the diagnostic executive cannot relocate the user script to another area. The diagnostic executive notifies you if you have violated this type of restriction by issuing a script incompatibility message.

- **Test number**
- **Run environment.** This defines the environment from which the actual diagnostic test can be run. The choices are 0 = ROM, 1 = MSI RAM, 2 = Main Memory, and 3 = Fastest Possible. Choose number 3 to select the fastest possible environment that will not overwrite the test.
- **Repeat code**
- **Error severity level**
- **Console error report**
- **Script error treatment**
- **LED display**
- **Console display**
- **Parameters**

Example 4-1 shows how to build and run a user script.

The utility displays the test name after you enter the test number, and the number of bytes remaining after you enter the information for each test. When you have finished entering tests, press **Return** at the next Next test number: prompt to end the script-building session. Then, type T A0 **Return** to run the new script.

You cannot review or edit a script you have created.

#### Example 4-1: Creating a Script with Utility 9F

```
SP=201406A8
Create script in ?[0=SSC, 1=Diag_RAM, 2=RAM] :1
Script starts at 2011FC00
1024 bytes left
Next test number :70
R3000_cache >>Run from ?[0=ROM, 1=Diag_RAM, 2=RAM, 3=fastest
possible] (0):3
R3000_cache >>Repeat? [0=no, 1=on error, 2=forever,
>2=count<FF] (0):0
R3000_cache >>Error severity? [0, 1, 2, 3] (2):2
R3000_cache >>Console error report? [0=none, 1=full] (1):1
R3000_cache >>Stop script on error? [0=NO, 1=YES] (1):1
R3000_cache >>LED on entry (07):
R3000_cache >>Console on entry (70):
1017 bytes left
Next test number :71
R3000_fpu >>Run from ?[0=ROM, 1=Diag_RAM, 2=RAM, 3=fastest
possible] (0):3
R3000_fpu >>Repeat? [0=no, 1=on error, 2=forever,
>2=count<FF] (0):0
R3000_fpu >>Error severity? [0, 1, 2, 3] (2):2
R3000_fpu >>Console error report? [0=none, 1=full] (1):1
R3000_fpu >>Stop script on error? [0=NO, 1=YES] (1):1
R3000_fpu >>LED on entry (07):
R3000_fpu >>Console on entry (71):
1010 bytes left
Next test number :A0 - script
1009 bytes left
Next test number :
>>> T A0
70..71..
>>>
```

Example 4-2 shows the script-building procedure to follow if (a) you are unsure of the test number to specify, and (b) you want to run one test repeatedly. If you are not sure of the test number, enter ? at the Next test number: prompt to invoke test 9E and display test numbers, test names, and so on. To run one test repeatedly, enter the following sequence:

1. Enter the test number (40 in Example 4-2) at the Next test number: prompt.
2. Enter A0 at the next Next test number: prompt.
3. Press **Return** at the next Next test number: prompt.
4. Enter T A0 to begin running the script repeatedly.
5. Press **CNC** to stop the test.

The sequence above is a useful alternative to using the REPEAT console command to run a test, because REPEAT (test) displays only line feeds; it does not display the console test announcement.

## Example 4-2: Listing and Repeating Tests with Utility 9F

```
>>>t 9f

SP=201406A8
Create script in ?[0=SSC, 1= Diag_RAM, 2=RAM] :0
Script starts at 201407E0
 24 bytes left
Next test number :?
Test
# Address Name Parameters
-----
2004C000 De_SCB
2004B200 CP_SCB
C6 2004D6DC SSC_powerup *****
C7 2004D79E CBTCR_timeout ***
34 2004D858 ROM_logic_test *

47 2005608B MEM_Refresh start_end_incr_cont_on_err_time_seconds *****
40 20056216 MEM_Count_Errs First_board_Last_board_Soft_errs_allowed *****
44 20056530 Cache_memory addr_incr *****
9D 2005687C Utilities Expnd_err_msg_get_mode_init_LEDs_clr_ps_cnt
9C 20056980 List_CPU_regs *
9F 20057195 Create_script *****
70 200578B8 R3000_cache *
71 200578CF R3000_fpu *
72 200578E4 R3000_tlb *
73 200579F9 R3000_reg_intrf loop_run_test ***
74 20057913 R3000_buf_intrf incr_pattern_start_add_end_add_run_test **
75 2005792D R3000_mem_intrf incr_pattern_start_add_end_add **
76 20057947 R3000_interrupt_run_test
77 20057961 R3000_mov_inver_incr * start_add_end_add **
78 20057C47 R3000_write_buf *
 24 bytes left
Next test number :40
MEM_Count_Errs>>Run from ?[0=ROM,1=Diag_RAM,3=fastest possible] (0):3
MEM_Count_Errs>>Repeat? [0=no,1=on_error,2=forever,>2=count<FF] (0):0
MEM_Count_Errs>>Error severity ? [0,1,2,3] (2):
MEM_Count_Errs>>Console error report? [0=none,1=full] (1):
MEM_Count_Errs>>Stop script on error? [0=NO,1=YES] (1):
MEM_Count_Errs>>LED on entry (04):
MEM_Count_Errs>>Console on entry (40):
MEM_Count_Errs>> First Board : 00000001 - 00000004 ?1
MEM_Count_Errs>> Last Board : 00000001 - 00000004 ?(00000004): 4
MEM_Count_Errs>> Soft_errs_allowed : 00000000 - FFFFFFFF ?(FFFFFFFF)
 5 Bytes left
Next test number :a0 - script
 4 bytes left
Next test number :
>>>t a0
40 4C..40..40..40..40..40..40..40..40..40..40..40..40..40..40..
40..40..40..40..
>>>
```

MLO-003808

### 4.3.5 Console Displays

Example 4-3 shows a typical console display during execution of the CVAX ROM-based diagnostics (power-up).

#### Example 4-3: Console Display (No Errors)

```
KN210-A Vn.n
Performing Normal System Tests
51..50..49..48..47..46..45..44..43..42..41..40..39..38..37..36..35..
34..33..32..31..30..29..28..27..26..25..24..23..22..21..20..19..18..
17..16..15..14..13..12..11..10..09..08..07..06..05..04..03..
Tests completed
>>>
```

The first line contains the module (KN210-A) and the firmware version (Vn.n).

The numbers on the console display do not refer to actual test numbers. Table 4-4 shows the actual ROM-based diagnostic tests that run during power-up.

**Table 4-4: Tests Run During Power-Up**

Number Displayed/Test Run	Number Displayed/Test Run	Number Displayed/Test Run
51 / 91	34 / C7	18 / C5
50 / 90	33 / 46	17 / 45
49 / 33	32 / C2	16 / 5A
48 / 32	31 / 4F	15 / 5C
47 / 50	30 / 4E	14 / 5D
46 / 5B	29 / 4D	13 / 5E
45 / 31	28 / 4C	12 / 5F
44 / 49	27 / 4B	11 / 72
43 / 30	26 / 4A	10 / 71
42 / 52	25 / 48	9 / 70
41 / 52	24 / 47	8 / 78
40 / 53	23 / 40	7 / 73
39 / C1	22 / 44	6 / 74
38 / 34	21 / 80	5 / 75
37 / C5	20 / 54	4 / 76
36 / 57	19 / 34	3 / 41
35 / 55		

During execution of the IPT, normal error messages are displayed if the console terminal is working. Console announcements, such as test numbers and countdown, however, are suppressed. Tests continue to run after the IPT, up to and including the appropriate console test.

Diagnostic test failures, if specified in the firmware script, produce error displays in the formats shown in Examples 4-4 and 4-5.

#### Example 4-4: Sample Output with Errors (CVAX)

```
746 2 07 FE 10 0002
P1=002F0000 P2=00000000 P3=00000000 P4=00FF0000 P5=00000000
P6=00000000 P7=00000000 P8=00000000 P9=00FF0000 P10=00000000
r0=00000000 r1=00010000 r2=55555555 r3=00000080 r4=AAAAAAAA
r5=00000080 r6=01EF0000 r7=20080144 r8=00010000 ERF=20140770
```

```
Normal operation not possible.
>>>
```

#### Example 4-5: Sample Output with Errors (R3000)

```
773 2 35 FE 00 0002
P1=00000100 P2=00000000 P3=00000000 P4=00000000 P5=00000000
P6=00000000 P7=00000000 P8=00000000 P9=00000000 P10=55555555
epc=BFC15850 sr=B0400000 badvaddr=00000000 cause=30002000
gp=00000000 sp=A0FF7BB0 fp=00000000
```

```
Normal operation not possible.
>>>
```

The errors are printed in a five-line display. The first line has six fields:

```
Test   Severity  Error   De_error  Vector  Count
```

- Test identifies the diagnostic test. In Example 4-4, the test is 46.
- Severity is the severity level of a test failure, as dictated by the script. In Example 4-4, 2 is the severity level. Failure of a severity level 2 test causes the display of this five-line error printout and halts an autoboot. An error of severity level 1 causes a display of the first line of the error printout but does not interrupt an autoboot. Most tests have a severity level of 2.



- Error is two hex digits identifying, usually within 10 instructions, where in the diagnostic the error occurred. This field is also called the subtestlog. In Example 4-4, 07 is the area where the error occurred.
- De\_error (diagnostic executive error) signals the diagnostic's state and any illegal behavior. This field indicates a condition that the diagnostic expects on detecting a failure. FE or EF in this field means that an unexpected exception or interrupt was detected. FF indicates an error as a result of normal testing, such as a miscompare. The possible codes are:
  - FF—Normal error exit from diagnostic
  - FE—Unanticipated interrupt (as in Example 4-4)
  - FD—Interrupt in cleanup routine
  - FC—Interrupt in interrupt handler
  - FB—Script requirements not met
  - FA—No such diagnostic
  - EF—Unanticipated exception in executive
- Vector identifies the SCB vector (10 in the example above) through which the unexpected exception or interrupt trapped, when the de\_error field detects an unexpected exception or interrupt (FE or EF).
- Count is four hex digits. It shows the number of previous errors that have occurred (two in Example 4-4).

Lines 2 and 3 of the error printout are parameters 1 through 10. When the diagnostics are running normally, these parameters are the same parameters that are listed in Table 4-1.

When an unexpected machine check exception or other type of exception occurs during the executive (de\_error is EF), the stack is saved in the parameters on lines 2 and 3, as listed in Tables 4-5 and 4-6.

**Table 4-5: Values Saved, Machine Check Exception During Executive (CVAX)**

Parameter	Value
P1	Contents of SP, points to vector value in P2
P2	Vector = 04, vector of exception 04-FC, 00 = Q-bus
P3	Address of PC pointing to failed instruction, P9
P4	Byte count = 10
P5	Machine check code
P6	Most recent virtual address
P7	Internal state information 1
P8	Internal state information 2
P9	PC, points to failing instruction
P10	PSL

**Table 4-6: Values Saved, Exception During Executive (CVAX)**

Parameter	Value
P1	Contents of SP, points to vector value in P2
P2	Vector = nn, vector of exception 04-FC, 00 = Q-bus
P3	Address of PC pointing to failed instruction, P4
P4	PC, points to instruction following failed instruction
P5	PSL
P6	Contents of stack
P7	Contents of stack
P8	Contents of stack
P9	Contents of stack
P10	Contents of stack

Lines 4 and 5 of the error printout are general registers R0 through R8 and the hardware error summary register.

When returning a module for repair, record the first line of the error printout and the version of the ROMs on the module repair tag.

Table 4-7 lists the hex LED display, the default action on errors, and the most likely FRUs. The table is divided into IPTs and scripts.

The Default on Error column refers to the action taken by the diagnostic executive under the following circumstances:

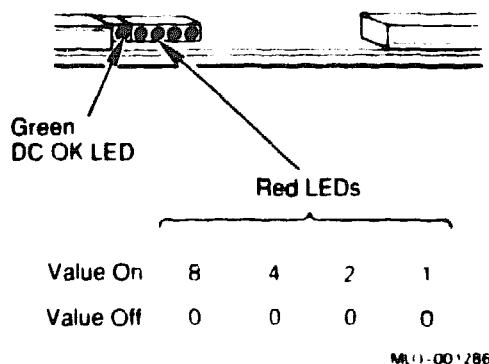
- The diagnostic executive detects an unexpected exception or interrupt.
- A test fails and that failure is reported to the diagnostic executive.

The Default on Error column does not refer to the action taken by the memory tests. The diagnostic executive either halts the script or continues execution at the next test in the script.

Most memory tests have a continue on error parameter (labeled `cont_on_error`, as shown in test 47 in Example 4-2). If you explicitly set `cont_on_error`, using parameter 4 in a memory test, the test marks bad pages in the bitmap and continues without notifying the diagnostic executive of the error. In this case, a halt on error does not occur even if you specify halt on error in the diagnostic executive (by answering Yes to Stop script on error? in Utility 9F), since the memory test does not notify the diagnostic executive that an error has occurred.

Figure 4-1 shows the LEDs on the KN210 CPU. They correspond to the hex display on the H3602-AB.

Figure 4-1: KN210 CPU Module LEDs



**Table 4-7: KN210 Console Displays and FRUs**

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU <sup>1</sup>
---------	------------------------	-----------------------	------------------	-------------	------------------

**Initial Power-Up Tests**

F	None	None	Loop on test	Power-up	7, 1, 5, 6
D	None	None	Loop on test	WAIT_POK	1
4	None	None	Loop on self	Entering IPT	1
7	None	None	Loop on test	SLU_EXT_LOOPBACK <sup>2</sup>	8, 9, 1

**Script BA**

C	None	79D	Continue	Utilities	1, 2
B	None	742	Continue	Check_for_intrs	1, 2
C	None	7C6	Continue	SSC_power-up	1, 2
7	None	760	Continue	CONSOLE_SERIAL	1, 2

End of script.

**Script AC**

Invoke script BC.

Invoke script A1.

End of script.

<sup>1</sup>In the case of multiple FRUs, refer to Section 4.5.2 for further information. If a problem recurs with the same FRU, check that the tolerance for system power supply +5 Vdc, +12 Vdc, and ac ripple are within specification.

<sup>2</sup>This test runs only if the power-up mode switch on the H3602-AB is set to the test position. See Section 4.6.3.

**FRU key:**

- 1 = KN210 CPU module
- 2 = KN210 I/O module
- 3 = MS650 module
- 4 = Memory interconnect cable
- 5 = Q22-bus device
- 6 = Q22/CD backplane
- 7 = System power supply
- 8 = H3602-AB CPU I/O panel
- 9 = H3602-AB cable
- 10 = CPU and I/O module interconnect cable

**Table 4-7 (Cont.): KN210 Console Displays and FRUs**

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU <sup>1</sup>
------------	------------------------------	-----------------------------	---------------------	-------------	------------------

### Script AA

Invoke script BA.

Invoke script BC.

Invoke script A1.

End of script.

### Script BC

8	51	791	Continue	CQBiC_power-up	1
8	50	790	Continue	CQBIC_registers	1
9	49	733	Continue	CMCTL_power-up	1
9	48	732	Continue	CMCTL_registers	1
6	47	750	Continue	M7636_present	10, 2, 1
6	46	75B	Continue	registers	1
9	45	731	Continue	MEM_setup_CSRs	1, 3, 4, 6
9	44	749	Continue	MEM_FDM_logic	3, 1, 4, 6
9	43	730	Halt	MEM_bitmap	3, 1, 4, 6

End of script.

### Script A1

Invoke script BD.

End of script.

<sup>1</sup>In the case of multiple FRUs, refer to Section 4.5.2 for further information. If a problem recurs with the same FRU, check that the tolerance for system power supply +5 Vdc, +12 Vdc, and ac ripple are within specification.

### FRU key:

- 1 = KN210 CPU module
- 2 = KN210 I/O module
- 3 = M8650 module
- 4 = Memory interconnect cable
- 5 = Q22-bus device
- 6 = Q22/CD backplane
- 7 = System power supply
- 8 = H3602-AB CPU I/O panel
- 9 = H3602-AB cable
- 10 = CPU and I/O module interconnect cable

**Table 4-7 (Cont.): KN210 Console Displays and FRUs**

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU <sup>1</sup>
<b>Script BD</b>					
C	42	752	Continue	PROG_TIMER_0	1
C	41	752	Continue	PROG_TIMER_1	1
C	40	753	Continue	TOY_CLOCK	1
C	39	7C1	Continue	SSC_RAM	1
B	38	734	Continue	ROM logic test	1
B	37	7C5	Continue	SSC_registers	1
6	36	757	Continue	SII_memory	1
B	35	755	Continue	INTERVAL_TIMER	1
C	34	7C7	Continue	CBTCR_timeout	1
B	33	746	Continue	CACHE1_DIAG_MODE	1
C	32	7C2	Continue	SSC_RAM_all	1
9	31	74F	Continue	MEM_data	3, 1, 4, 6
9	30	74E	Continue	MEM_byte	3, 1, 4, 6
9	29	74D	Continue	MEM_addr	3, 1, 4, 6
9	28	74C	Continue	MEM_ECC_error	3, 1, 4, 6
9	27	74B	Continue	MEM_masked_errors	3, 1, 4, 6
9	26	74A	Continue	MEM_correction	3, 1, 4, 6
9	25	748	Continue	MEM_address_shorts	3, 1, 4, 6
9	24	747	Continue	MEM_refresh	3, 1, 4, 6
9	23	740	Continue	MEM_count_errs	3, 1, 4, 6
B	22	744	Continue	CACHE1_MEMORY	1, 3, 4, 6
8	21	780	Continue	CQBIC_MEMORY	1, 3, 5, 4, 6
B	20	754	Continue	VIRTUAL_MODE	1, 3, 4, 6

<sup>1</sup>In the case of multiple FRUs, refer to Section 4.5.2 for further information. If a problem recurs with the same FRU, check that the tolerance for system power supply +5 Vdc, +12 Vdc, and ac ripple are within specification.

**FRU key:**

- 1 = KN210 CPU module
- 2 = KN210 I/O module
- 3 = MS650 module
- 4 = Memory interconnect cable
- 5 = Q22-bus device
- 6 = Q22/CD backplane
- 7 = System power supply
- 8 = H3602-AB CPU I/O panel
- 9 = H3602-AB cable
- 10 = CPU and I/O module interconnect cable

**Table 4-7 (Cont.): KN210 Console Displays and FRUs**

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU <sup>1</sup>
<b>Script BD</b>					
(Continued from previous page.)					
C	19	734	Continue	ROM_logic	1
C	18	7C5	Continue	SCC_registers	1
8	17	745	Continue	CACHE_MEM_CQBIC	1, 3, 5, 4, 6
9	16	75A	Continue	VAX CMCTL CDAL	1
6	15	75C	Continue	SII_initiator	2, 1
6	14	75D	Continue	SII_target	2, 1
5	13	75E	Continue	NI_memory	2, 1
5	12	75F	Continue	NI_test	2, 1
A	11	772	Continue	R3000_tlb	1
A	10	771	Continue	R3000_fpu	1
A	09	770	Continue	R3000_cache	1
A	08	778	Continue	R3000_write_buf	1
A	07	773	Continue	R3000_reg_intrf	1, 2
A	06	774	Continue	R3000_buf_intrf	2, 1
A	05	775	Continue	R3000_mem_intrf	3, 1, 4, 6
A	04	776	Continue	R3000_interrupt	1, 2
C	03	741	Continue	Board reset	1, 5
End of script.					

**Script A9**

9	4F	74F	Halt	MEM_data	3, 1, 4, 6
9	4E	74E	Halt	MEM_byte	3, 1, 4, 6

<sup>1</sup>In the case of multiple FRUs, refer to Section 4.5.2 for further information. If a problem recurs with the same FRU, check that the tolerance for system power supply +5 Vdc, +12 Vdc, and ac ripple are within specification.

**FRU key:**

- 1 = KN210 CPU module
- 2 = KN210 I/O module
- 3 = MS650 module
- 4 = Memory interconnect cable
- 5 = Q22-bus device
- 6 = Q22/CD backplane
- 7 = System power supply
- 8 = H3602-AB CPU I/O panel
- 9 = H3602-AB cable
- 10 = CPU and I/O module interconnect cable

**Table 4-7 (Cont.): KN210 Console Displays and FRUs**

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU <sup>1</sup>
<b>Script A9</b>					
(Continued from previous page.)					
9	4D	74D	Halt	MEM_dds	3, 1, 4, 6
9	4C	74C	Halt	MEM_ECC_error	3, 1, 4, 6
9	4B	74B	Halt	MEM_masked_errors	3, 1, 4, 6
9	4A	74A	Halt	MEM_correction	3, 1, 4, 6
9	48	748	Continue	MEM_Addr_shrts	3, 1, 4, 6
9	47	747	Continue	MEM_refresh	3, 1, 4, 6
9	40	740	Continue	MEM_count_errs	3, 1, 4, 6
C	41	741	Continue	Board reset	1, 5

End of script.

**Script A8**

9	31	731	Halt	CMCTL_setup_CSRs	1, 3, 4, 6
9	49	749	Halt	MEM_FDM_logic	3, 1, 4, 6
9	30	730	Halt	MEM_bitmap	1, 3, 4, 6

Invoke script A7.

End of script.

**Script A7**

9	4F	74F	Halt	MEM_data	3, 1, 4, 6
9	4E	74E	Halt	MEM_byte	3, 1, 4, 6

<sup>1</sup>In the case of multiple FRUs, refer to Section 4.5.2 for further information. If a problem recurs with the same FRU, check that the tolerance for system power supply +5 Vdc, +12 Vdc, and ac ripple are within specification.

**FRU key:**

- 1 = KN210 CPU module
- 2 = KN210 I/O module
- 3 = MS650 module
- 4 = Memory interconnect cable
- 5 = Q22-bus device
- 6 = Q22/CD backplane
- 7 = System power supply
- 8 = H3602-AB CPU I/O panel
- 9 = H3602-AB cable
- 10 = CPU and I/O module interconnect cable



**Table 4-7 (Cont.): KN210 Console Displays and FRUs**

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU <sup>1</sup>
<b>Script A7</b>					
(Continued from previous page.)					
9	4D	74D	Halt	MEM_addr	3, 1, 4, 6
9	4C	74C	Halt	MEM_ECC_error	3, 1, 4, 6
9	4B	74B	Halt	MEM_masked_errors	3, 1, 4, 6
9	4A	74A	Halt	MEM_correction	3, 1, 4, 6
9	48	748	Halt	MEM_address_shorts	3, 1, 4, 6
9	47	747	Halt	MEM_refresh	3, 1, 4, 6
9	40	740	Continue	MEM_count_bad pages	3, 1, 4, 6
8	80	780	Continue	CQBIC_memory	1, 3, 5, 4, 6
C	41	741	Halt	Board reset	1, 5
End of script.					

**Script BE**

A	72	772	Halt	R3000_tlb	1
A	71	771	Halt	R3000_fpu	1
A	70	770	Halt	R3000_cache	1
A	78	778	Halt	R3000_write_buf	1
6	50	750	Continue	M7636_present	10, 2, 1
A	73	773	Halt	R3000_reg_intrf	2, 1
A	74	774	Halt	R3000_buf_intrf	1, 2
A	75	775	Halt	R3000_mem_intrf	3, 1, 4, 6
A	76	776	Halt	R3000_interrupt	1, 2
A	77	777	Halt	R3000_mov_inver	1, 2

<sup>1</sup>In the case of multiple FRUs, refer to Section 4.5.2 for further information. If a problem recurs with the same FRU, check that the tolerance for system power supply +5 Vdc, +12 Vdc, and ac ripple are within specification.

**FRU key:**

- 1 = KN210 CPU module
- 2 = KN210 I/O module
- 3 = MS850 module
- 4 = Memory interconnect cable
- 5 = Q22-bus device
- 6 = Q22/CD backplane
- 7 = System power supply
- 8 = H3602-AB CPU I/O panel
- 9 = H3602-AB cable
- 10 = CPU and I/O module interconnect cable

**Table 4-7 (Cont.): KN210 Console Displays and FRUs**

Hex LED	Normal Console Display	Error Console Display	Default on Error	Description	FRU <sup>1</sup>
------------	---------------------------	--------------------------	---------------------	-------------	------------------

**Script BE**

(Continued from previous page.)

End of script.

**Script BF**

6	50	750	Continue	M7636_present	10, 2, 1
6	5B	75B	Halt	SII_registers	2, 1
6	57	757	Halt	SII_memory	2, 1
6	5C	75C	Halt	SII_initiator	2, 1
6	5D	75D	Halt	SII_target	2, 1
6	5E	75E	Halt	NI_memory	2, 1
6	5F	75F	Halt	NI_test	2, 1

End of script.

<sup>1</sup>In the case of multiple FRUs, refer to Section 4.5.2 for further information. If a problem recurs with the same FRU, check that the tolerance for system power supply +5 Vdc, +12 Vdc, and ac ripple are within specification.

**FRU key:**

- 1 = KN210 CPU module
- 2 = KN210 I/O module
- 3 = MS650 module
- 4 = Memory interconnect cable
- 5 = Q22-bus device
- 6 = Q22/CD backplane
- 7 = System power supply
- 8 = H3602-AB CPU I/O panel
- 9 = H3602-AB cable
- 10 = CPU and I/O module interconnect cable

### 4.3.6 System Halt Messages

Table 4-8 lists messages that may appear on the console terminal when a system error occurs.

**Table 4-8: System Halt Messages**

Code	Message	Explanation
702	EXT HLT	External halt, caused either by console BREAK condition or because Q22-bus BHALT_L or DBR<AUX_HLT> bit was set while enabled.
703	-	Power-up; no halt message displayed. The presence of the firmware banner and diagnostic countdown indicates this halt.
704	ISP ERR	Caused by attempt to push interrupt or exception state onto the interrupt stack when the interrupt stack was mapped NO ACCESS or NOT VALID.
705	DBL ERR	A second machine check occurred while the processor was attempting to service a normal exception.
706	HLT INST	The processor executed a HALT instruction in kernel mode.
707	SCB ERR3	The vector had bits <1:0> = 3.
708	SCB ERR2	The vector had bits <1:0> = 2.
70A	CHM FR ISTK	A change mode instruction was executed when PSL<IS> was set.
70B	CHM TO ISTK	The SCB vector for a change mode had bit <0> set.
70C	SCB RD ERR	A hard memory error occurred during a processor read of an exception or interrupt vector.
710	MCHK AV	An access violation or an invalid translation occurred during machine check exception processing.
711	KSP AV	An access violation or an invalid translation occurred during invalid kernel stack pointer exception processing.
712	DBL ERR2	Double machine check error. A machine check occurred during an attempt to service a machine check.
713	DBL ERR3	Double machine check error. A machine check occurred during an attempt to service a kernel stack not valid exception.
719	PSL EXC5	PSL <26:24> = 5 on interrupt or exception.
71A	PSL EXC6	PSL <26:24> = 6 on interrupt or exception.
71B	PSL EXC7	PSL <26:24> = 7 on interrupt or exception.
71D	PSL REI5	PSL <26:24> = 5 on an rei instruction.
71E	PSL REI6	PSL <26:24> = 6 on an rei instruction.
71F	PSL REI7	PSL <26:24> = 7 on an rei instruction.

### 4.3.7 Console Error Messages

Table 4–9 lists messages issued in response to an error or to a console command that was entered incorrectly.

**Table 4–9: Console Error Messages**

Code	Message	Explanation
720	CORRPTN	The console data base was corrupted. The console simulates a power-up sequence and rebuilds its data base.
721	ILL REF	The requested reference would violate virtual memory protection, address is not mapped or is invalid in the specified address space, or value is invalid in the specified destination.
722	ILL CMD	The command string cannot be parsed.
723	INV DGT	A number has an invalid digit.
724	LTL	The command was too large for the console to buffer. The message is sent only after the console receives the <b>Return</b> at the end of the command.
725	ILL ADR	The specified address is not in the address space.
726	VAL TOO LRG	The specified value does not fit in the destination.
727	SW CONF	Switch conflict. For example, an EXAMINE command specifies two different data sizes.
728	UNK SW	The switch is not recognized.
729	UNK SYM	The EXAMINE or DEPOSIT symbolic address is not recognized.
72A	CHKSM	An X command has an incorrect command or data checksum. If the data checksum is incorrect, this message is issued and is not abbreviated to "Illegal command."
72B	HLTED	The operator entered a HALT command.
72C	FND ERR	A FIND command failed either to find the RPB or 64 Kbytes of good memory.
72D	TMOUT	Data failed to arrive in the expected time during an X command.
72E	MEM ERR	Memory error or machine check occurred.
72F	UNKINT	An unexpected interrupt or exception occurred.
730	UNIMPLEMENTED	Unimplemented function.
731	QUAL NOVAL	Qualifier does not take a value.
732	QUAL AMBG	Ambiguous qualifier.
733	QUAL REQ VAL	Qualifier requires a value.
734	QUAL OVERF	Too many qualifiers.
735	ARG OVERF	Too many arguments.
736	AMBG CMD	Ambiguous command.
737	INSUF ARG	Too few arguments.

### 4.3.8 VMB Error Messages (CVAX)

If VMB is unable to boot MDM, it returns an error message to the console. Table 4-10 lists the error messages and their descriptions.

**Table 4-10: VMB Error Messages**

Message Number	Mnemonic	Interpretation
740	NOSUCHDEV	No bootable devices found
741	DEVASSIGN	Device is not present
742	NOSUCHFILE	Program image not found
743	FILESTRUCT	Invalid boot device file structure
744	BADCHKSUM	Bad checksum on header file
745	BADFILEHDR	Bad file header
746	BADDIRECTORY	Bad directory file
747	FILNOTCNTC	Invalid program image format
748	ENDOFFILE	Premature end-of-file encountered
749	BADFILENAME	Bad file name given
74A	BUFFEROVF	Program image does not fit in available memory
74B	CTRLERR	Boot device I/O error
74C	DEVINACT	Failed to initialize boot device
74D	DEVOFFLINE	Device is off line
74E	MEMERR	Memory initialization error
74F	SCBINT	Unexpected SCB exception or machine check
750	SCB2NDINT	Unexpected exception after starting program image
751	NOROM	No valid ROM image found
752	NOSUCHNODE	No response from load server
753	INSFMAPREG	Insufficient Q-bus mapping registers due to invalid memory configuration, bad memory, or because Q-bus map was not relocated to main memory
754	RETRY	No devices bootable, retrying

## 4.4 Acceptance Testing

Perform the acceptance testing procedure listed below, after installing a system or whenever replacing the following:

KN210 CPU module  
KN210 I/O module  
MS650-BA memory module  
Memory data interconnect cable  
Backplane  
DSSI device  
H3602-AB

1. Run five error-free passes of the power-up scripts by entering the following command:

```
>>> R T 0
```

Press **Ctrl/C** to terminate the scripts.

2. Enter the following command to run script A1, which invokes CPU and memory tests without resetting the bitmap to mark only solid multibit ECC errors in main memory. This command gives you a quick memory check, since most tests run on a 256-Kbyte boundary.

```
>>> T A1
```

Perform the next two steps for a more thorough test of memory.

```
>>> T A8  
>>> R T A7  
>>> R T BE
```

The first command runs script A8 for one pass. This command enables mapping out of solid single-bit ECC as well as multibit ECC errors. It will also run script A7 for one pass.

The second command runs script A7 repeatedly. This command runs the memory tests only and does not reset the bitmap. Press **CM/C** after two passes to terminate the script. This test takes up to 5 minutes per pass, depending on the amount of memory in the system. Most of the memory diagnostics test memory on a page boundary.

The third command runs script BE repeatedly. This command runs a series of R3000 tests. Press **CM/C** after five passes to terminate the script.

If any of the memory tests fails, it marks the bitmap and continues with no error printout to the console. An exception is test 40 (count bad pages). If any single-bit or multibit ECC errors are detected, they are reported in test 40. Such a failure indicates that pages in memory have been marked bad in the bitmap because of solid single-bit and/or multibit ECC errors. The error printout does not display the 20 longwords, since it is a severity level 1 error.

3. Check the memory configuration again, since test 31 can check for only a few invalid configurations. For example, test 31 cannot report that a memory board is missing from the configuration, since it has no way of knowing if the board should be there or not.

To check the memory configuration, enter the following command line:

```
>>> SHOW MEMORY/FULL
```

```
Memory 0: 00000000 to 00FFFFFF, 16MB, 0 bad pages
```

```
Total of 16MB, 0 bad pages, 104 reserved pages
```

```
Memory Bitmap
```

```
-00FF3000 to 00FF3FFF, 8 pages
```

```
Console Scratch Area
```

```
-00FF4000 to 00FF7FFF, 32 pages
```

```
Qbus Map
```

```
-00FF8000 to 00FFFFFF, 64 pages
```

```
Scan of Bad Pages
```

```
>>>
```

Memories 0 through 3 are the MS650 memory modules. The Q22-bus map always spans the top 32 Kbytes of good memory. The memory bitmap always spans two pages (1 Kbyte) per 4 Mbytes of memory configured.

Use utility 9C to compare the contents of configuration registers MEMCSR 0–15 with the memory installed in the system:

```
>>> T 9c
TOY  =96C972E5  ICCS =00000000
TCR0 =00000000  TIR0 =00000000  TNIR0=00000000  TIVR0=00000078
TCR1 =80000084  TIR1 =00000000  TNIR1=00000000  TIVR1=0000007C
RXCS =00000000  RXDB =00000011  TXCS =00000000  TXDB =00000030
MSER =00000000  CADR =0000000C
BDR  =FFFFFFE2  DLEDR=0000000B  SSCCR=00D46677  CBTCR=00000004
SCR  =0000D000  DSER =00000000  QBEAR=0000000F  DEAR =00000000
QBMBR=00000000  IPCRn=0000

MEM_FRU1 MEMCSR_0=80000017  1=80400017  2=80800017  3=80C00017
MEM_FRU2 MEMCSR_4=81000017  5=81400017  6=81800017  7=81C00017
MEM_FRU3 MEMCSR_8=82000017  9=82400017  10=82800017  11=82C00017
MEM_FRU4 MEMCSR12=83000017  13=83400017  14=83800017  15=83C00017
          MEMCSR16=8190000F  17=00002000

Ethernet  SA = 08-00-2B-0F-8D-C2  NICSR0=0004

SII      MSIDR0 =0000  MSIDR1 =0000  MSIDR2 =0000  MSICSR =0000
          MSIDR  =0007  MSITR  =0000  MSITLP =0000  MSIILP =0000
          MSIDSCR=0000  MSIDSSR=A480  MSIDCR =0000

>>>
```

One memory bank is enabled for each 4 Mbytes of memory. The MEMCSRs map modules as follows:

MEMCSR 0–3	First MS650 memory module
MEMCSR 4–7	Second MS650 memory module
MEMCSR 8–11	Third MS650 memory module
MEMCSR 12–15	Fourth MS650 memory module



Verify the following:

- The bank enable bits are set in all four MEMCSRs. MEMCSRs <6:0> should equal 17 hex for all four MEMCSRs. See the values for MEMCSR 0-3 in the example.
  - If a memory board is not present, bits <31:0> are all zeros for the corresponding group of four MEMCSRs. See the values for MEMCSR 8-11 in the example.
  - Bits <25:22> should always increment by one starting at zero in MEMCSR0 if bit <31> equals 1. In the example above, bits <25:22> of MEMCSR 0 through 7 increment by one, resulting in an increment in each word. If bit <31> equals 0, <25:22> should equal zero.
4. Check the Q22-bus and the Q22-bus logic in the KN210 CQBIC chip, and the configuration of the Q22-bus, as follows:

>>> SHOW QBUS

Scan of Qbus I/O Space

```
-20000120 (760440) = 0080 (300) DHQ11/DHV11/CXA16/CXB16/CXY08
-20000122 (760442) = F081
-20000124 (760444) = DD18
-20000126 (760446) = 0200
-20000128 (760450) = 0000
-2000012A (760452) = 0000
-2000012C (760454) = 8000
-2000012E (760456) = 0000
-20001920 (774440) = FF08 (120) DELQA/DEQNA
-20001922 (774442) = FF00
-20001924 (774444) = FF2B
-20001926 (774446) = FF06
-20001928 (774450) = FF16
-2000192A (774452) = FFF2
-2000192C (774454) = 00F8
-2000192E (774456) = 1030
-20001940 (774500) = 0000 (260) TQK50/TQK70/TU81E/RV20/K-TAPE
-20001942 (774502) = 0BC0
-20001F40 (777500) = 0020 (004) IPCR
```

Scan of Qbus Memory Space

>>>

The columns are described below. The examples listed are from the last line of the example above.

First column = the VAX I/O address of the CSR, in hex (20001F40).

Second column = the Q22-bus address of the CSR, in octal (777500).

Third column = the data, contained at the CSR address, in hex (0020).

Fourth column = the device vector in octal, according to the fixed or floating Q22-bus and UNIBUS algorithm (004).

Fifth column = the device name (IPCR, the KN210 interprocessor communications register).

Additional lines for the device are displayed if more than one CSR exists.

The last line, Scan of Qbus Memory Space, displays memory residing on the Q22-bus, if present. VAX memory mapped by the Q22-bus map is not displayed.

If the system contains an MSCP or TMSCP controller, run test 81. This test performs the following functions:

- Performs step one of the UQ port initialization sequence

- Performs the SA wraparound test

- Checks the Q22-bus interrupt logic

If you do not specify the CSR address, the test searches for and runs on the first MSCP device by default. To test the first TMSCP device, you must specify the first parameter:

```
>>> T 81 20001940
```

You can specify other addresses if you have multiple MSCP or TMSCP devices in the first parameter. This action may be useful to isolate a problem with a controller, the KN210, or the backplane. Use the VAX address provided by the SHOW QBUS command to determine the CSR value. If you do not specify a value, the MSCP device at address 20001468 is tested by default.

5. Check that all UQSSP, MSCP, TMSCP, and Ethernet controllers and devices are visible by typing the following command line:

```
>>> SHOW DEVICE
DSSI Node 0 (R3WB0A)
-DIA0 -rf(0,0,*) (RF71)

DSSI Node 1 (R3QDVA)
-DIA1 -rf(1,1,*) (RF71)

DSSI Node 7 (*)

UQSSP Tape Controller 0 (774500)
-MUA0 -tm(0,0) (TK70)

Ethernet Adapter
-ESA0 -se -mop() (08-00-2B-0C-C4-75)
```

In the example above, the console displays the DSSI node names and numbers, then the CVAX device names and the R3000 devices names, which are followed by the controller number and unit number in parentheses.

DSSI Node 7 (\*) is the DSSI adapter located on the KN210 I/O module. In most cases, the KN210 I/O module is the local DSSI node shown by the asterisk and has a node number of 7. DSSI node names, node numbers, and unit numbers should be unique.

The UQSSP (TQK70) tape controller and its CSR address are also shown. The line below this display shows a TK70 connected.

The next two lines show the logical name and station address for the Ethernet adapter located on the KN210 I/O module.

6. Test the DSSI subsystem using the KN210 ROM-based Diagnostics and Utilities Protocol (DUP) facility. This facility allows you to connect to the DUP server in the RF drive controller. Here are some examples:

```
>>> SET HOST/DUP/DSSI 7
Starting DUP server...

Stopping DUP server...
```

In this example, a DUP connection was made with DSSI node 7, the KN210 I/O module. The DUP server times out, since no local programs exist and no response packet was received.

>>> SET HOST/DUP/DSSI 1  
Starting DUP server...

DSSI Node 1 (R3VBNC)  
DRVEXR V1.0 D 21-FEB-1988 21:27:54  
DRVSTST V1.0 D 21-FEB-1988 21:27:54  
HISTORY V1.0 D 21-FEB-1988 21:27:54  
ERASE V1.0 D 21-FEB-1988 21:27:54  
PARAMS V1.0 D 21-FEB-1988 21:27:54  
DIRECT V1.0 D 21-FEB-1988 21:27:54  
End of directory

Task Name? DRVSTST

Write/read anywhere on medium? (1=Yes/(0=No)): <CR>

5 minutes for test to complete.

Compare failed on head 1 track 1091.

Compare failed on head 0 track 529.

Task Name? DRVEXR

Write/read anywhere on medium? (1=Yes/(0=No)): <CR>

Test time in minutes? ((10)-100):

10 minutes for test to complete.

R3VBNC::MSCP\$DUP 21-FEB-1988 21:37:35 DRVEXR CPU=00:00:01.88 PI=43

R3VBNC::MSCP\$DUP 21-FEB-1988 21:37:38 DRVEXR CPU=00:00:03.38 PI=79

Compare failed on head 1 track 1091.

R3VBNC::MSCP\$DUP 21-FEB-1988 21:37:40 DRVEXR CPU=00:00:04.97 PI=116

^C

>>>

In the example above, the local programs DRVSTST and DRVEXR are run on drive 1. Do not enter 1 in response to the question Write/read anywhere on medium?. Doing so will destroy the data on the disk. Press **Return**, which uses the default, allowing reads and writes to the DBNs only. **CMT** or **CWC** displays a message as shown in the DRVEXR example above (the lines beginning with R3VBNC::). In the example, **CMT** has been pressed twice to show the difference in the time and in the value of the progress indicator (PI).

Press **CWC** to terminate the program

Use the local programs HISTORY (Section 4.8.3) and PARAMS (Section 4.8.5) to determine the cause of errors displayed during DRVSTST or DRVEXR. DRVSTST should run successfully for one pass on each drive. Customer Services can refer to the *RF71 Disk Drive Service Manual* for details about the DUP local programs and corrective action.

7. If there are one or more DELQA modules in the system, use test 82 to invoke the Ethernet option's self-test and receive status from the host firmware. Test 82 is useful for acceptance testing if you cannot access the system enclosure to see the DELQA LEDs.

8. After the steps above have completed successfully, load MDM and run the system tests from the Main Menu. If they run successfully, the system has gone through its basic checkout and you can load the software.

## 4.5 Troubleshooting

This section contains suggestions for determining the cause of ROM-based diagnostic test failures.

### 4.5.1 FE Utility

If any of the IPT tests fail, the major test code is displayed on the LEDs. After the IPT tests are completed, any test that fails is displayed on both the LEDs and through a console display. Run the FE utility if the message, Normal operation not possible, is displayed after the tests have completed and there is no other error indication, or if you need more information than what is provided in the error display.

The FE utility dumps diagnostic state to the console (Example 4-6). This state indicates the major and minor test codes of the test that failed, the 10 parameters associated with the test, and the hardware error summary register.

#### Example 4-6: FE Utility Example

```
>>> T FE
bitmap=00BF3400, length=0C00, checksum=007E
busmap=00BF8000
return_stack=201406A8
subtest_pc=2004F4C4
timeout=00000001, error=0B, de_error=FE
de_error_vector=18, severity_code=02, total_error_count=0001
previous_error=FE0B5D5D, 00000000, 00000000, 00000000, 00000000
last_exception_pc=20050807
flags=01FFFD7F, test_flags=20
highest_severity=00
led_display=05
console_display=5D
save_mchk_code=80, save_err_flags=000000
param_1=00000100 2=00000100 3=000000F7 4=00000000 5=00000001
param_6=00000004 7=20050527 8=00000000 9=20140698 10=200521F4
```

The most useful fields displayed above are as follows:

- **De\_error\_vector**, which is the SCB vector through which the unexpected interrupt or exception trapped if de\_error equals FE or EF.
- **Total\_error\_count**. Four hex digits showing the number of previous errors that have occurred.
- **Previous\_error**. Contains the history of the last four errors. Each longword contains four bytes of information. From left to right these are the de\_error, subtest\_log, and test number (copied in both bytes).
- **Save machine check code (save\_mchk\_code)**. Valid only if the test halts on error. This field has the same format as the hardware error summary register.
- **Save error flags (save\_err\_flags)**. Valid only if the test halts on error. This field has the same format as the hardware error summary register.
- **Parameters 1 through 10**. Valid only if the test halts on error. The parameters have the same format as the hardware error summary register.

FE in the previous\_error field indicates that an unexpected exception has occurred. If any of the tests that announce to the console fails, and the error code is FE, examine the last longword of the error printout. The last longword is the hardware error summary register and contains the machine check code (<31:24>) and KN210 error status bits (<23:0>). Table 4-11 lists the status bits.

**Table 4-11: Hardware Error Summary Register**

Bit	Register	Description
31	Machine check code	
30	Machine check code	
29	Machine check code	
28	Machine check code	
27	Machine check code	
26	Machine check code	
25	Machine check code	
24	Machine check code	
23	MSER <6>	; CDAL data parity error
22	MSER <5>	; Mchn chck CDAL parity error
21	MSER <4>	; Machine check cache parity
20	MSER <1>	; Cache data parity error
19	MSER <0>	; Cache tag parity error
18	Unused	
17	MEMCSR16 <31>	; Uncorrectable ECC error
16	MEMCSR16 <30>	; Two or more uncorrectable errors
15	MEMCSR16 <29>	; Correctable single-bit error
14	MEMCSR16 <25>	; Page address bits 25:22 of
13	MEMCSR16 <24>	; Location that caused error.
12	MEMCSR16 <23>	; These four bits point to the
11	MEMCSR16 <22>	; failing 4-Mbyte bank of memory.
10	MEMCSR16 <8>	; DMA read/write error.
9	MEMCSR16 <7>	; CDAL parity error on write.
8	CBCTR <31>	; CDAL bus timeout.
7	CBCTR <30>	; CPU read/write bus timeout.
6	DSER <7>	; Q22-bus NXM.
4	DSER <5>	; Q22-bus parity error.
3	DSER <4>	; Read main memory error.
2	DSER <3>	; Lost error.
1	DSER <2>	; No grant timeout.
0	IPCRn <15>	; DMA Q22-bus memory error.

## 4.5.2 Isolating Memory Failures

This section describes procedures for isolating memory subsystem failures, particularly when the system contains more than one MS650 memory module.

### 1. SHOW MEMORY/FULL

Use the SHOW MEMORY/FULL command to examine failures detected by the memory tests. Use this command if test 40 fails, which indicates that pages have been marked bad in the bitmap.

You can also use SHOW MEMORY/FULL after terminating a script that is taking an unusually long time to run. Press **[CWC]** to terminate the script after the completion of the current test. (**[CWC]** on the KN210 console takes effect only after the entire script completes.) After terminating the script, enter SHOW MEMORY/FULL to see if the tests have marked any pages bad up to that point. See Section 4.4 for an example of this command.

### 2. T A9

```
>>> T [memory test] starting_board_number ending_board_number
```

Script A9 runs only the memory tests and halts on the first error detected. Unlike the power-up script, it does not continue. Since the script does not rerun the test, it detects memory-related failures that are not hard errors. You should then run the individual test that failed on each memory module one MS650 module at a time. You can input parameters 1 and 2 of tests 40, 47, 48, and 4A through 4F as the starting and ending address for testing. It is easier, however, to input the memory module numbers 1–4. For example, if test 4F fails, test the second memory module as follows:

```
>>> T 4F 2 2
```

If a failure is detected for a second of three MS650 modules, for example, repeat this procedure for all memory modules to isolate the MS650 module that is the FRU, using the process of elimination.

You can also specify the address increment. For example, to test the third memory module on each page boundary, type:

```
>>> T 4F 3 3 200
```

By default, most memory tests test one longword on a 256-Kbyte boundary. If an error is detected, the tests start testing on a page boundary. Test 48 (address shorts test) is an exception: it checks every location in memory since it can do so in a reasonable amount of time. Other tests, such as 4F (floating ones and zeros test), can take up to



one hour, depending on the amount of memory, if each location were to be tested. If you do specify an address increment, do not input less than 200 (testing on a page boundary), since almost all hard memory failures span at least one page. For normal servicing, do not specify the address increment, since it adds unnecessary repair time; most memory subsystem failures can be found using the default parameter.

All the memory tests, with the exception of 40, save MEMCSR17 and MEMCSR16 memory status and error registers in parameters 7 and 8, respectively.

### 3. T 9C

The utility 9C is useful if test 31 or some other memory test failed because memory was not configured correctly.

To help in isolating an FRU, examine registers MEMCSR 0-15 by entering T 9C at the console I/O mode prompt (Example 4-7). Utility 9C is also useful for examining the error registers MSER, DSER, and MEMCSR16, upon a fatal system crash or similar event.

### 4. T 40

Although the SHOW MEMORY command displays pages that are marked bad by the memory test and is easier to interpret than test 40, there is one instance in which test 40 reports information that SHOW MEMORY does not report. You can use test 40 as an alternative to running script A9 to detect soft memory errors. Specify the third parameter in test 40 (see Table 4-1) to be the threshold for soft errors. To allow 0 (zero) errors, enter the following:

```
>>> T 40 1 4 0
```

This command tests the memory on the four memory modules. Use it after running memory tests individually or within a script. If test 40 fails with subtestlog = 6, examine longwords P5-P8 to determine how many errors have been detected on memory modules one through four, as follows:

Longword	Memory Module Number
P5	1
P6	2
P7	3
P8	4

#### Example 4-7: Isolating Bad Memory Using T 9C

74F SCBINT

>>>t 9C

```
TOY =96C972E5 ICCS =00000000
TCR0 =00000000 TIRO =00000000 TNIRO=00000000 TIVRO=00000078
TCR1 =80000084 TIR1 =00000000 TNIR1=00000000 TIVR1=0000007C
RXCS =00000000 RXDB =00000011 TXCS =00000000 TXDB =00000030
MSER =00000000 CADR =0000000C
BDR =FFFFFFE2 DLEDR=0000000B SSCCR=00D46677 CBTCR=00000004
SCR =0000D000 DSER =00000000 QBEAR=0000000F DEAR =00000000
QBMBR=00000000 IPCRn=0000
```

```
MEM_FRU1 MEMCSR_0=80000017 1=80400017 2=80800017 3=80C00017
MEM_FRU2 MEMCSR_4=81000017 5=81400017 6=81800017 7=81C00017
MEM_FRU3 MEMCSR_8=82000017 9=82400017 10=82800017 11=82C00017
MEM_FRU4 MEMCSR12=83000017 13=83400017 14=83800017 15=83C00017
MEMCSR16=8190000F 17=00002000
```

```
Ethernet SA = 08-00-2B-0F-8D-C2 NICSRO=0004
```

```
SII MSIDR0 =0000 MSIDR1 =0000 MSIDR2 =0000 MSICSR =0000
MSIIDR =0007 MSITR =0000 MSITLP =0000 MSIILP =0000
MSIDSCR=0000 MSIDSSR=A480 MSIDCR =0000
```

>>>

In this case, the diagnostics have passed but MDM does not boot. One of the console/VMB error messages is displayed. Run utility 9C and examine the error registers. Bit 31 in MEMCSR 16 is set, indicating an uncorrectable ECC error in memory. If any of bits <31:29> is set, there was a memory error. Compare the bits MEMCSR16 <25:22> against MEMCSR 0-15 <25:22>. If they match and the MEMCSRn <31> is set, then the board that was experiencing the failure (the memory FRU) is the MEM\_FRU number on the left.

In Example 4-7, the FRU is the second memory FRU, which is the second MS650-BA module because both conditions are met by

MEMCSR\_6 in the MEM\_FRU 2 row. The following conditions are shown in Example 4-8:

- MEMCSR\_6 matches MEMCSR16, since bits <25:22> (Bank Number) match.
- The Bank Enable bit <31> in MEMCSR\_6 is set, which indicates that the bank number is valid.

#### Example 4-8: 9C—Conditions for Determining a Memory FRU

	3	2	2	
	1	5	2	
MEMCSR16 = 8194000F Hex =	1000	0001	1001	0100 0000 0000 0000 1111
MEMCSR_6 = 81800017 Hex =	1000	0001	1000	0000 0000 0000 0001 0111
	^		^	
	bit 31 set		25:22 match	

### 4.5.3 Additional Troubleshooting Suggestions

Note the following additional suggestions when diagnosing a possible memory failure.

If more than one memory module is failing, you should suspect the KN210 CPU module, KN210 I/O module, CPU/memory cable, backplane, or MS650 modules as the cause of failure.

Always check the seating of the memory cable first before replacing a KN210 or MS650 module. If the seating appears to be improper, rerun the tests. Also remember to leave the middle connector disconnected for a three-connector cable when the system is configured with only one MS650.

If you are rotating MS650 modules to verify that a particular memory module is causing the failure, be aware that a module may fail in a different way when in a different slot.

Be sure to put the modules back in their original positions when you are finished.

If memory errors are found in the error log, use the KN210 ROM-based diagnostics to see if it is an MS650 problem or if it is related to the KN210, CPU/memory interconnect cable, or backplane. Follow steps 1-3 of Section 4.4 and Section 4.5.2 to aid in isolating the failure.

Use the SHOW QBUS, SHOW DEVICE, and SET HOST/DUP commands when troubleshooting I/O subsystem problems.

Use the CONFIG command to help with configuration problems or when installing new options onto the Q-bus. See the command descriptions in Chapter 3.

You can run a DSSI device power-up diagnostic without performing a cold restart or spinning the disk drives down and back up.

Type the following at the console program:

```
>>> T 58 <node_number>
```

A CI Reset command is issued to the DSSI device, causing the device to perform its power-up diagnostics.

Parameter 1 is the DSSI node or port number. It must be in the range of 0–7 (0 is the default). Use the default for parameter 2.

You can perform this test repeatedly with the REPEAT command (R T 58 <node\_number>). In that case the drive's self-tests run repeatedly until you press **Ctrl/C** to terminate the test.

Once the test has completed successfully, you can examine the DSSI device's internal error logs by running the DUP local programs HISTRY and PARAMS. Refer to Section 4.8.3 and Section 4.8.5 for further information.

## 4.6 Loopback Tests

You can use external loopback tests to localize problems with the Ethernet, console, and DSSI subsystems.

### 4.6.1 DSSI Problems

For DSSI problems, run the SII external loopback test (test 56). To check the DSSI bus out to the KN210 I/O module connector, plug one end of the cable (17-02216-01) to the H3281 loopback connector and the other end to the DSSI connector on the KN210 I/O module. To test out to the end of the DSSI bus, power down the system, remove all DSSI devices with the exception of the KN210 from the bus, and plug the external DSSI loopback connector 12-30702-01 in place of the DSSI bus terminator.

### 4.6.2 Ethernet Problems

For Ethernet problems, run the external loopback test (NI test, number 5F) by entering the following:

```
>>> T 5F 1<CR>
```

Set parameter 1 to run this test. Only the external loopback test runs. Be sure to set the ThinWire/standard Ethernet switch on the H3602-AB to the appropriate position.

Use two 50-ohm H8225 terminators connected to an H8223 T-connector. Before running the test, attach this assembly to the H3602-AB ThinWire port.

To test the standard Ethernet connector, attach loopback connector 12-22196-02 to the H3602-AB standard Ethernet port.

### **4.6.3 Testing the Console Port**

To test the console port at power-up, set the operation switch on the H3602-AB, using the procedure in Section 3.4.3. The H3103 connects the console port transmit and receive lines. At power-up, the SLU\_EXT\_LOOPBACK IPT then runs a continuous loopback test.

To test the end of the console terminal cable:

1. Plug the MMJ end of the console terminal cable into the H3602-AB.
2. Disconnect the other end of the cable from the terminal.
3. Plug an H8572 adapter into the disconnected end of the cable.
4. Connect the H3103 to the H8572.

While the test is running, the LED display on the CPU I/O insert should alternate between 7 and 4. A value of 7 latched in the display indicates a test failure. If the test fails, one of the following parts is faulty: the KN210, the H3602-AB, the cabling, or the CPU module.

## **4.7 Module Self-Tests**

Module self-tests run when you power up the system. A module self-test can detect hard or repeatable errors, but usually not intermittent errors.

Module LEDs display pass/fail test results.

A pass by a module self-test does not guarantee that the module is good, because the test usually checks only the controller logic. The test usually does not check the module Q22-bus interface, the line drivers and receivers, or the connector pins—all of which have relatively high failure rates.

A fail by a module self-test is accurate, because the test does not require any other part of the system to be working.

The following modules do not have LED self-test indicators:

KLESI  
LPV11  
TSV05

The following modules have one green LED, which indicates that the module is receiving +5 and +12 Vdc:

CXA16  
CXB16  
CXY08

Table 4-12 lists loopback connectors for common KN210 system modules. Refer to *Microsystems Options* for a description of specific module self-tests.

**Table 4-12: Loopback Connectors for Q22-bus Devices**

Device	Module Loopback	Cable Loopback
CXA16/CXB16	H3103 + H8572 <sup>1</sup>	
CXY08	H3046 (50-pin)	H3197 (25-pin)
DELQA	12-22196-02	
KN210/H3602-AB	H3103	H3103 + H8572

<sup>1</sup>For DSSI to KN210 or RF-series connector, use 17-02216-01 plus H3281 loopback. For connection to end of bus, use the DSSI loopback connector 12-30702-01.

## 4.8 RF-Series ISE Troubleshooting and Diagnostics

An RF-series integrated storage element (ISE) may fail either during initial power-up or during normal operation. In both cases, the failure is indicated by the lighting of the red fault LED on the OCP on the enclosure front panel. The ISE also has a red fault LED, but it is not visible from the outside of the system enclosure.

If the drive is unable to execute the Power-On Self-Test (POST) successfully, the red fault LED remains lit and the ready LED does not come on, or both LEDs remain on.

POST is also used to handle two types of error conditions in the drive:

1. *Controller errors* are caused by the hardware associated with the controller function of the drive module. A controller error is fatal to the operation of the drive, since the controller cannot establish a logical connection to the host. If the red fault LED remains lit, replace the drive module.

2. *Drive errors* are caused by the hardware associated with the drive control function of the drive module. These errors are not fatal to the drive, since the drive can establish a logical connection and report the error to the host. Both LEDs go out for about 1 second, then the red fault LED lights. In this case, run either DRVSTST, DRVEXR, or PARAMS (described in the next sections) to determine the error code.

Three configuration errors also commonly occur:

- More than one node with the same node number
- Identical node names
- Identical unit numbers

The first error cannot be detected by software. Use the SHOW DSSI command to display the second and third errors. This command lists each device connected to the DSSI bus by node name and unit number.

If the ISE is connected to the OCP, you must install a unit ID plug in the corresponding socket on the OCP. If the ISE is not connected to the OCP, the ISE reads its unit ID from the three-switch DIP switchpack on the side of the drive.

The RF-series ISE contains the following local programs (described in the following sections):

DIRECT	A directory, in DUP specified format, of available local programs
DRVSTST	A comprehensive drive functionality verification test
DRVEXR	A utility that exercises the ISE
HISTRY	A utility that saves information retained by the drive
ERASE	A utility that erases all user data from the disk
PARAMS	A utility that allows you to look at or change drive status, history, and parameters

A description of each local program follows, including a table showing the dialog of each program. The table also indicates the type of messages contained in the dialog, although the screen display will not indicate the message type. Message types are abbreviated as follows:

Q—Question  
I—Information  
T—Termination  
FE—Fatal error

To access these local programs, use the console SET HOST/DUP command, which creates a virtual terminal connection to the storage device and the designated local program, using the Diagnostic and Utilities Protocol (DUP) standard dialog.

Once the connection is established, the local program is in control. When the program terminates, control is returned to the KN210 console. To abort or prematurely terminate a program and return control to the KN210 console, press **CM/C** or **CM/Y**.

### 4.8.1 DRVTST

DRVTST is a comprehensive functionality test. Errors detected by this test are isolated to the FRU level. The messages are listed in Table 4-13.

**Table 4-13: DRVTST Messages**

Message Type	Message
I	Copyright © 1988 Digital Equipment Corporation
Q	Write/read anywhere on the medium? (1=yes/0=no)
Q	User data will be corrupted. Proceed? (1=yes/0=no)
I	Five minutes to complete.
T	Test passed.
Or	
FE	Unit is currently in use <sup>1</sup>
FE	Operation aborted by user.
FE	XXXX—Unit diagnostics failed. <sup>2</sup>
FE	XXXX—Unit read/write test failed <sup>2</sup>

<sup>1</sup>Either the drive is inoperative, in use by a host, or is currently running another local program.

<sup>2</sup>Refer to the diagnostic error code list at the end of this chapter.

Answering No to the first question ("Write/read...?") or pressing **Return** results in a read-only test. DRVTST, however, writes to a diagnostic area on the disk. Answering Yes to the first question or pressing **Return** causes the second question to be displayed.

Answering No to the second question ("Proceed?") or pressing **Return** is the same as answering No to the first question. Answering Yes to the second question permits write and read operations anywhere on the medium.

DRVTST resets the ECC error counters, then calls the timed I/O routine. After the timed I/O routine ends (5 minutes), DRVTST saves the counters again. It computes the uncorrectable error rate and byte (symbol) error rate. If either rate is too high, the test fails and the appropriate error code is displayed.



## 4.8.2 DRVEXR

The DRVEXR local program exercises the ISE. The test is data transfer intensive and indicates the overall integrity of the device. Table 4-14 lists the DRVEXR messages.

**Table 4-14: DRVEXR Messages**

Message Type	Message
I	Copyright © 1988 Digital Equipment Corporation
Q	Write/read anywhere on the medium? (1=yes/0=no)
Q	User data will be corrupted. Proceed? (1=yes/0=no)
Q	Test time in minutes? ((10)-100)
I	ddd minutes to complete
I	ddddddddd blocks (512 bytes) transferred.
I	ddddddddd bytes in error (soft).
I	ddddddddd uncorrectable ECC errors (recoverable).
T	Complete.

Or:

FE	Unit is currently in use. <sup>1</sup>
FE	Operation aborted by user.
FE	XXXX—Unit diagnostics failed. <sup>2</sup>
FE	XXXX—Unit read/write test failed <sup>2</sup>

<sup>1</sup> Either the drive is inoperative, in use by a host, or is currently running another local program.

<sup>2</sup> Refer to the diagnostic error list at the end of this chapter.

Answering No to the first question ("Write/read...?") results in a read-only test. DRVEXR, however, writes to a diagnostic area on the disk. Answering Yes to the first question causes the second question to be displayed.

Answering No to the second question ("Proceed?") is the same as answering No to the first question. Answering Yes to the second question permits write and read operations anywhere on the medium.

**NOTE:** If the write-protect switch on the OCP is pressed in (LED on) and you answer Yes to the second question, the drive does not allow the test to run. DRVEXR displays the error message 2006—Unit read/write test failed. In this case, the test has not failed but has been prevented from running.

DRVEXR saves the error counters, then calls the timed I/O routine. After the timed I/O routine ends, DRVEXR saves the counters again. It then reports the total number of blocks transferred, bits in error, bytes in error, and uncorrectable errors.

DRVEXR uses the same timed I/O routine as DRVTST, with two exceptions:

- DRVTST always uses a fixed time of five minutes, whereas you specify the time of the DRVEXR routine
- DRVTST determines whether the drive is good or bad. DRVEXR reports the data but does not determine the condition of the drive.

### 4.8.3 HISTRY

The HISTRY local program displays information about the history of the ISE. Table 4-15 lists the HISTRY messages.

**Table 4-15: HISTRY Messages**

Message Type	Field Length	Field Meaning
I	47 ASCII characters	Copyright notice
I	4 ASCII characters	Product name
I	12 ASCII characters	Drive serial number
I	6 ASCII characters	Node name
I	1 ASCII character	Allocation class
I	8 ASCII characters	Firmware revision level
I	17 ASCII characters	Hardware revision level
I	6 ASCII characters	Power-on hours
I	5 ASCII characters	Power cycles
I <sup>1</sup>	4 ASCII characters	Hexadecimal fault code
T		Complete

<sup>1</sup>Displays the last 11 fault codes as informational messages. Refer to the diagnostic error code list at the end of this chapter.

The following example shows a typical screen display when you run HISTORY.

```
Copyright © 1988 Digital Equipment Corporation
RF71
EN01082
SUSAN
0
RFX V101
RF71 PCB-5/ECO-00
617
21
A04F
A04F
A103
A04F
A404
A04F
A404
A04F
A404
A04F
A404
Complete.
```

If no errors have been logged, no hexadecimal fault codes are displayed.

#### 4.8.4 ERASE

The ERASE local program overwrites application data on the drive while leaving the replacement control table (RCT) intact. This local program is used if an HDA must be replaced and the customer wants to protect any confidential or sensitive data.

Use ERASE only if the HDA must be replaced and only after you have backed up the customer's data.

Table 4-16 lists the ERASE messages.

**Table 4-16: ERASE Messages**

Message Type	Message
I	Copyright © 1988 Digital Equipment Corporation
Q	Write/read anywhere on the medium? (1=yes/0=no)
Q	User data will be corrupted. Proceed? (1=yes/0=no)
I	6 minutes to complete.
T	Complete.

Or:

FE	Unit is currently in use.
FE	Operation aborted by user.
FE	xxxx—Unit diagnostics failed. <sup>1</sup>
FE	xxxx—Operation failed. <sup>2</sup>

<sup>1</sup>Refer to the diagnostic error code list at the end of this chapter.

<sup>2</sup>xxxx = one of the following error codes:

000D : Cannot write the RCT.

000E : Cannot read the RCT.

000F : Cannot find an RBN to revector to.

0010 : The RAM copy of the bad block table is full.

If a failure is detected, the message indicating the failure will be followed by one or more messages containing error codes.

## 4.8.5 PARAMS

The PARAMS local program supports modifications to device parameters that you may need to change, such as device node name and allocation class. You invoke it in the same way as the other local programs. However, you use the following commands to make the modifications you need:

EXIT	Terminates PARAMS program
HELP	Prints a brief list of commands and their syntax
SET	Sets a parameter to a value
SHOW	Displays a parameter or a class of parameters
STATUS	Displays module configuration, history, or current counters, depending on the status type chosen
WRITE	Alters the device parameters

#### 4.8.5.1 EXIT

In maintenance mode, use the EXIT command to terminate the PARAMS local program.

#### 4.8.5.2 HELP

In maintenance mode, use the HELP command to display a brief list of available PARAMS commands, as shown in the example below.

```
PARAMS> HELP
EXIT
HELP
SET {parameter | .} value
SHOW {parameter | . | ,class}
    /ALL      /CONST  /DRIVE
    /SERVO    /SCS     /MSCP
    /DUP
STATUS {type}
    CONFIG  LOGS      DATALINK
    PATHS
WRITE
PARAMS>
```

#### 4.8.5.3 SET

In maintenance mode, use the SET command to change the value of a given parameter. *Parameter* is the name or abbreviation of the parameter to be changed. *Value* is the value assigned to the parameter.

For example, SET NODE SUSAN sets the NODENAME parameter to SUSAN.

The following parameters are useful to Customer Services:

ALLCLASS	The controller allocation class. The allocation class should be set to match that of the host.
FIVEDIME	True (1) if MSCP should support five connections with ten credits each. False (0) if MSCP should support seven connections with seven credits each.
UNITNUM	The MSCP unit number.
FORCEUNI	True (1) if the unit number should be taken from the DSSI ID. False (0) if the UNITNUM value should be used instead.
NODENAME	The controller's SCS node name.
FORCENAM	True (1) if the SCS node name should be forced to the string RF71x (where x is a letter from A to H corresponding to the DSSI bus ID) instead of using the NODENAME value. False (0) if NODENAME is to be used.

#### 4.8.5.4 SHOW

Use the **SHOW** command to display the settings of a parameter or a class of parameters. The **SHOW** command displays the full name of the parameter (8 characters or less), the current value, the default value, radix and type, and any flags associated with each parameter.

#### 4.8.5.5 STATUS

Use the **STATUS** command to display module configuration, history, or current counters, depending on the type specified. *Type* is the optional ASCII string that denotes the type of display desired. If you omit *Type*, all available status information is displayed. If present, it may be abbreviated. The following types are available:

<b>CONFIG</b>	Displays the module name, node name, power-on hours, power cycles, and other such configuration information. Unit failures are also displayed, if applicable.
<b>LOGS</b>	Displays the last 11 machine and bug checks on the module. The display includes the processor registers (D0-D7, A0-A7), the time and date of each failure (if available; otherwise the date 17 November 1858 is displayed), and some of the hardware registers.
<b>DATALINK</b>	Displays the data link counters.
<b>PATHS</b>	Displays available path information (open virtual circuits) from the point of view of the controller. The display includes the remote node names, DSSI IDs, software type and version, and counters for the messages and datagrams sent and/or received.

#### 4.8.5.6 WRITE

Use the **WRITE** command to write the changes made while in **PARAMS** to the drive nonvolatile memory. The **WRITE** command is similar to the **VMS SYSGEN WRITE** command. Parameters are not available, but you must be aware of the system and/or drive requirements and use the **WRITE** command accordingly or it may not succeed in writing the changes.

The **WRITE** command may fail for one of the following reasons:

- You altered a parameter that required the unit, and the unit cannot be acquired (that is, the unit is not available to the host). Changing the unit number is an example of a parameter that requires the unit.
- You altered a parameter that required a controller initialization, and you replied negatively to the request for reboot. Changing the node name or the allocation class are examples of parameters that require controller initialization.
- Initial drive calibrations were in progress on the unit. The use of the **WRITE** command is inhibited while these calibrations are running.

## 4.8.6 Diagnostic Error Codes

Diagnostic error codes appear when you are running DRVST, DRVEXR, or PARAMS. Most of the error codes indicate a failure of the drive module. The exceptions are listed below. The error codes are listed in Table 4-17. If you see any error code other than those listed below, replace the module.

**Table 4-17: RF-Series ISE Diagnostic Error Codes**

Code	Message	Meaning
2032/A032	Failed to see FLT go away	FLT bit of the spindle control status register was asserted for one of the following reasons: 1. Reference clock not present 2. Stuck rotor 3. Bad connection between HDA and module
203A/A03A	Cannot spin up, ACLOW is set in WrtFlt	Did not see ACOK signal, which is supplied by the host system power supply for staggered spin-up.
1314/9314	Front panel is broken	Could be either the module or the operator control panel or both.





## Appendix A

# ULTRIX-32 Exerciser and Uerf Command Summary

---

This appendix contains a summary of ULTRIX-32 exerciser and uerf commands to help you troubleshoot and diagnose errors in the DECsystem 5400.

See the following documents for detailed information on the commands:

- *ULTRIX-32 Guide to System Exercisers*
- *ULTRIX-32 Guide to the Error Logger System*

### A.1 Online ULTRIX Exerciser

The ULTRIX exercisers perform functional system and device testing. The exercisers are run in single- or multiuser mode from an account with *root* privileges.

The exercisers log status information in *LOG* files. Normal device errors are handled by the error log and *uerf*. You can run each of the exercisers in the background by ending each command line with an ampersand (&). This allows many (the same or different) exercisers to be run concurrently, which enhances your ability to perform system testing.

To run the exercisers, your current directory must be the *field* account. To terminate the exercisers, enter `CM/C` if the job is in the foreground, or `k111 -15 pid` if in the background. When you run an exerciser in the background, the *pid* is displayed when the command is invoked.

A time stamp entry is made in the system error log each time you stop or start an exerciser. Use the *uerf* option `-r 350` to include these in an error report. All the system exercisers, except *netx*, have the `-o` option. The `-o` option allows you to specify a file where diagnostic output is saved when the exerciser terminates.

#### Exercising More Than One Part of the System

You can run more than one exerciser at the same time. Keep in mind, however, that the more processes you have running, the slower the system

performs. Before exercising the system extensively, make sure there are no other users on the system.

To exercise more than one part of the system simultaneously, use the **syscript** maintenance command. The **syscript** command asks you which exercisers you want to run, how long you want to run each exerciser, and how many exercisers you want to run at one time. The **syscript** command allows you to exercise a device, a subsystem, or the entire system.

You can start each exerciser by using either of the following methods:

- Manually, by specifying the time parameter (-t option) and by placing each command in the background before executing the next command
- By typing the **syscript** command as follows:

```
# syscript
```

Once the **syscript** command is running, answer the questions displayed on the console. The **syscript** command then executes the individual exercisers and creates a file called **testsuite**, which contains all the answers you entered. You can reexecute the commands in the **testsuite** file by entering the following, which causes **testsuite** to execute using the original commands and parameters that you specified:

```
# sh testsuite
```

### A.1.1 Communications Exerciser (Asynchronous Serial Lines)

The communications exerciser writes, reads, and validates random data and packet lengths on communication lines as specified.

#### Syntax

```
cmx[-h][-ofile][-tmin]-lline#
```

#### Options

<b>-h</b>	Prints a help message.
<b>-ofile</b>	Writes run-time statistics to <i>file</i> . Default file is <b>#LOG_CMX_##</b> .
<b>-tmin</b>	Runs the exerciser for <i>x</i> minutes. Default is run continuously.
<b>-lline#</b>	Specifies the line number to exercise. For example, if the line to be exercised is <b>/dev/tty03</b> , <b>line#=03</b> .

## Usage

Any line to be exercised **MUST** have a loopback connector on the communication option's bulkhead panel or the end of the cable. Any line to be exercised **MUST** be disabled in the `/etc/ttys` file by setting the *status* to off.

## EXAMPLE:

Exercise line tty01 and tty03 for 10 minutes in the background:

```
cmx -t10 -l01 03
```

## A.1.2 Disk Exerciser

**CAUTION:** *This exerciser can **DESTRUCTIVELY WRITE** on a disk. Do not use this exerciser on any portion of a disk that contains customer data.*

*The -p and -c options destroy data on a disk. The -rdev command does not overwrite data.*

## Syntax

```
dskx[options]-rdev  
dskx[options]-pdevpart  
dskx[options]-cdev
```

## Arguments

<b>-rdev</b>	Random read-only test on all but the c partition
<b>-pdevpart</b>	Writes, reads, and validates on device <i>dev</i> on partition <i>part</i> .
<b>-cdev</b>	Writes, reads, and validates on device <i>dev</i> on all but the c partition.

## Options

<b>-h</b>	Prints a help message.
<b>-ofile</b>	Writes run-time statistics to <i>file</i> . Default file is <code>#LOG DSKX ##</code> .
<b>-tm</b>	Runs the exerciser for <i>m</i> minutes. Default is run continuously.

## EXAMPLE:

Test (read only) the first RA disk in the system (ra0) for 20 minutes in the background. Diagnostics display every 5 minutes:

```
dskx -rra0 -t20 -d5 &
```

### A.1.3 File System Exerciser

The file system exerciser initiates multiple processes and creates, writes, closes, opens, and reads a test file of random data.

#### Syntax

**fsx**[-h][-ofile][-tmin]-fpath[-pn]

#### Options

- h** Print a help message.
- ofile** Writes run-time statistics to *file*. Default file is *#LOG FSX ##*.
- tmin** Runs the exerciser for *x* minutes. Default is run continuously.
- fpath** Path name of the file system directory to test. Default is */usr/field*.
- pn** Number of *fsx* processes to spawn. Maximum is 250. Default is 20.

#### Usage

This test writes and reads data on the disk; it is not destructive to the customer's data. The file system exerciser can also be used on an NFS-mounted file system.

#### EXAMPLE:

Exercise the */usr/tmp* file system continuously using 10 processes in the background:

**fsx -p10 -f/usr/tmp &**

### A.1.4 Line Printer Exerciser

#### Syntax

**lpz**[-h][-ofile][-tmin]-fpath[-pn]

#### Arguments

- ddev** Printer device name to exercise.

#### Options

- h** Print a help message.
- ofile** Writes run-time statistics to *file*. Default file is *#LOG LPX ##*.

- tmin**                Runs the exerciser for *x* minutes. Default is run continuously.
- pn**                To save paper, pauses printing for *n* minutes and only exercises the controller. Default is 15. A value of 0 indicates no pause.

#### EXAMPLE:

Exercise lp1: lpx -dlp1

### A.1.5 Memory Exerciser

#### Syntax

**memx[-h][-s][-ofile][-tmin][-mj][-pt]**

#### Options

- h**                Prints a help message.
- s**                Disables shared memory testing. Shared memory is software functionality, not hardware.
- ofile**           Writes run-time statistics to *file*. Default file is #LOG\_MEMX\_##.
- tmin**            Runs exerciser for *x* minutes. Default is run continuously.
- mj**              Memory size in *j* bytes to be tested by each spawned process. Default is (total memory)/20.
- pt**              Number of *memx* processes to spawn. Maximum is 20. Default is 20.

#### Usage

The memory exerciser is restricted by available swap space. Errors like out of memory generally indicate swap space was used up. If you have more physical memory than swap space, you may see this problem. If so, reduce the number of spawned processes and/or the size of memory you are testing. Running the memory exerciser can also cause other users to have the same memory problem.

#### EXAMPLE:

Exercise all memory and the shared memory functionality for 10 minutes in the background:

**memx -t10 &**

## A.1.6 Magtape Exerciser

The magtape exerciser reads, writes, and validates random data from the beginning of the tape (BOT) to the end of the tape (EOT).

### Syntax

```
mtx[options]-adev  
mtx[options]-sdev  
mtx[options]-ldev  
mtx[options]-vdev
```

### Arguments

- adev**                Use short-, long-, and variable-length record tests on raw device *dev*.
- sdev**               Use short records on raw *dev*.
- ldev**               Use long records on raw *dev*.
- vdev**               Use variable records on raw *dev*.

### Options

- h**                   Prints a help message.
- ofile**              Writes run-time statistics to *file*. Default file is #LOG\_MTX\_##.
- ti**                   Runs exerciser for *i* minutes. Default is run continuously.
- rl**                   Record length for long record test. Range is 1 to 20480. Default is 10240.
- tk**                   Size of file in *k* number of records. Default: -1, go to EOT.

### EXAMPLE:

Run all record lengths on tape drive rmt0h for 5 minutes in the background:

```
mtx -armt0h -t5 &
```

## A.1.7 TCP/IP Network Exerciser

### Syntax

**netx[-h][-tmin][-pm]nodename**

### Arguments

**nodename**            Node name of target system to test. May also be the host system name

### Options

**-h**                    Prints a help message.

**-tmin**                Runs exerciser for x minutes. Default is run continuously.

**-pm**                  Port number.

### Usage

The TCP echo service defined in the */etc/inetd.conf* file must not be commented out (# at start of line) on the host and target systems.

### EXAMPLE:

Exercise the network from the local host to the remote node **max** continuously in the background:

**netx max &**

## A.2 Uerf Error Log Commands

The Uerf utility generates error log reports and does bit-to-text translation for hardware device registers and messages, similar to *ERF* on VMS. Syntax is case sensitive. If no options are specified, all errors are reported.

To disable error logging to an error log file, type:

```
# /etc/eli -d
```

To enable error logging in multiuser mode, type:

```
# /etc/eli -e
```

### Syntax

```
/etc/uerf[options...]
```

### Options

<b>-A adapter type</b>	<b>EXAMPLE: /etc/uerf -A uba,nmi</b>
<b>alo</b>	BVP controller
<b>alo</b>	BVP controller
<b>bla</b>	BI LESI adapter
<b>bus</b>	BI UNIBUS adapter
<b>nmi</b>	NMI errors
<b>uba</b>	VAX UNIBUS adapter
<b>default</b>	Report all error types

---

<b>-c classes</b>	<b>EXAMPLE: /etc/uerf -c oper</b>
<b>err</b>	All hardware and software errors
<b>maint</b>	Maintenance events
<b>oper</b>	System status; startup/shutdown; configuration

---



**-D** Reports errors for MSCP disks (*ra*, *rd*). Default: all MSCP disks are reported.  
**EXAMPLE:** `/etc/uerf -D ra60`

**-f** Specifies the error log file to be used to generate the report.  
**EXAMPLE:** `/etc/uerf -f old.errorlog`

**-h** Displays a brief help message.

**-H** Selects errors only for the specified system name.  
**EXAMPLE:** `/etc/uerf -H guru`

---

**-M mainframe\_errors** **EXAMPLE:** `/etc/uerf -M mem`

**cpu** Reports CPU errors and machine checks.

**mem** Reports memory errors (SBE and DBE).

**default** Reports all error types.

---

**-n** Uerf runs. Waits for errors to be logged and immediately reports them.

---

**-o output** **EXAMPLE:** `/etc/uerf -o full`

**brief** Reports errors in brief format (*default*).

**full** Reports all information for each error.

**tores** No bit-to-text translation for register values.

---

**-O operating\_system\_events** **EXAMPLE:** `/etc/uerf -O seg.ref`

**aef** Arithmetic exception faults

**aet** Asynchronous trap exception faults

**bpt** Breakpoint instruction faults

**cmp** Compatibility mode faults

**pag** Page faults

<b>pif</b>	<b>Privileged instruction faults</b>
<b>pro</b>	<b>Protection faults</b>
<b>ptf</b>	<b>Page table faults</b>
<b>raf</b>	<b>Reserved address faults</b>
<b>rof</b>	<b>Reserved operand faults</b>
<b>scf</b>	<b>System call exception faults</b>
<b>sog</b>	<b>Segmentation faults</b>
<b>tra</b>	<b>Trace exception faults</b>
<b>xfc</b>	<b>Reports xfc instruction faults</b>

---

**-R**                      **Reports errors in reverse chronological order.**

---

**-r record\_type**                      **EXAMPLE: /etc/uerf -r 102,210,250**

**Hardware Detected Error Types:**

<b>100</b>	<b>Machine check</b>
<b>101</b>	<b>Memory CRD/RDS errors</b>
<b>102</b>	<b>Disk errors</b>
<b>103</b>	<b>Tape errors</b>
<b>104</b>	<b>Device controller errors</b>
<b>105</b>	<b>Adapter errors</b>
<b>106</b>	<b>Bus errors</b>
<b>107</b>	<b>Stray interrupts</b>
<b>108</b>	<b>Asynchronous write errors</b>
<b>109</b>	<b>Exceptions/faults</b>
<b>112</b>	<b>Stack dump</b>

**Software Detected Error Types:**

<b>200</b>	<b>Panics (bug checks)</b>
<b>201</b>	<b>CI ppd information</b>

**Informational ASCII Message Types:**

<b>250</b>	<b>Informational</b>
------------	----------------------

**Operational Message Types:**

<b>300</b>	<b>Startup</b>
<b>301</b>	<b>Shutdown</b>
<b>310</b>	<b>Time change</b>
<b>350</b>	<b>Diagnostic information</b>
<b>351</b>	<b>Repair information</b>

---

**-s sequence numbers**                      Reports errors for the specified sequence numbers.  
**EXAMPLE:** /etc/uerf -s 1011,1320

---

**-S**    Summarizes error information.  
**EXAMPLE:** /etc/uerf -S -o full

---

**-t s:dd-mmm-yyyy, hh:mm:ss e:dd-mmm-yyyy, hh:mm:ss**  
**EXAMPLE:** /etc/uerf -t s:08-aug-1989:13:20:00

**s**    Starting date and time

**e**    Ending date and time

**dd**                                        Day

**mmm**                                      Month

**yyyy**                                    Year

hh	Hour
mm	Minute
ss	Second

---

**-T** Reports errors for TMSCP tapes (*th*, *tu*). Default: all TMSCP tapes are reported.  
**EXAMPLE:** /etc/uerf -T tu81

---

**-x** Excludes specified error types from the report.  
**EXAMPLE:** /etc/uerf -x -r 102,103

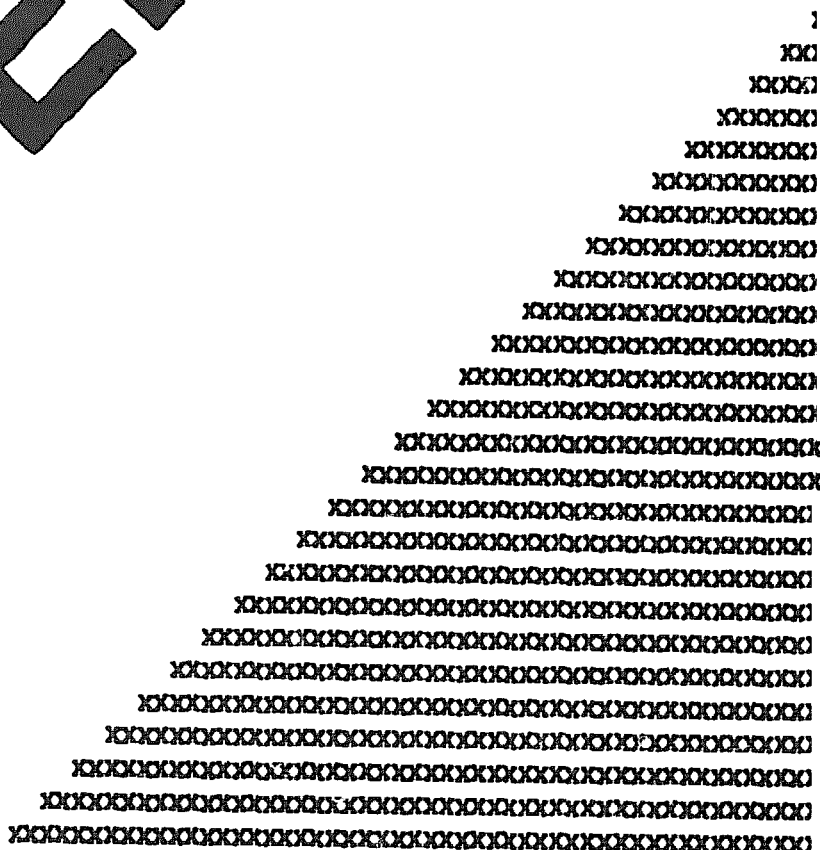
---

**-Z** Displays the entire error record as hexadecimal data. Used only for debugging.

---

[illegible]

APPENDIX B



## Appendix B

# KN210 Address Assignments

---

This appendix explains how to access R3000 physical address locations and provides physical address space maps for the KN210 CPU module set.

### B.1 Accessing Physical Locations (R3000)

From the R3000 processor, you must use virtual addresses to access physical locations. The R3000 processor accepts virtual addresses only.

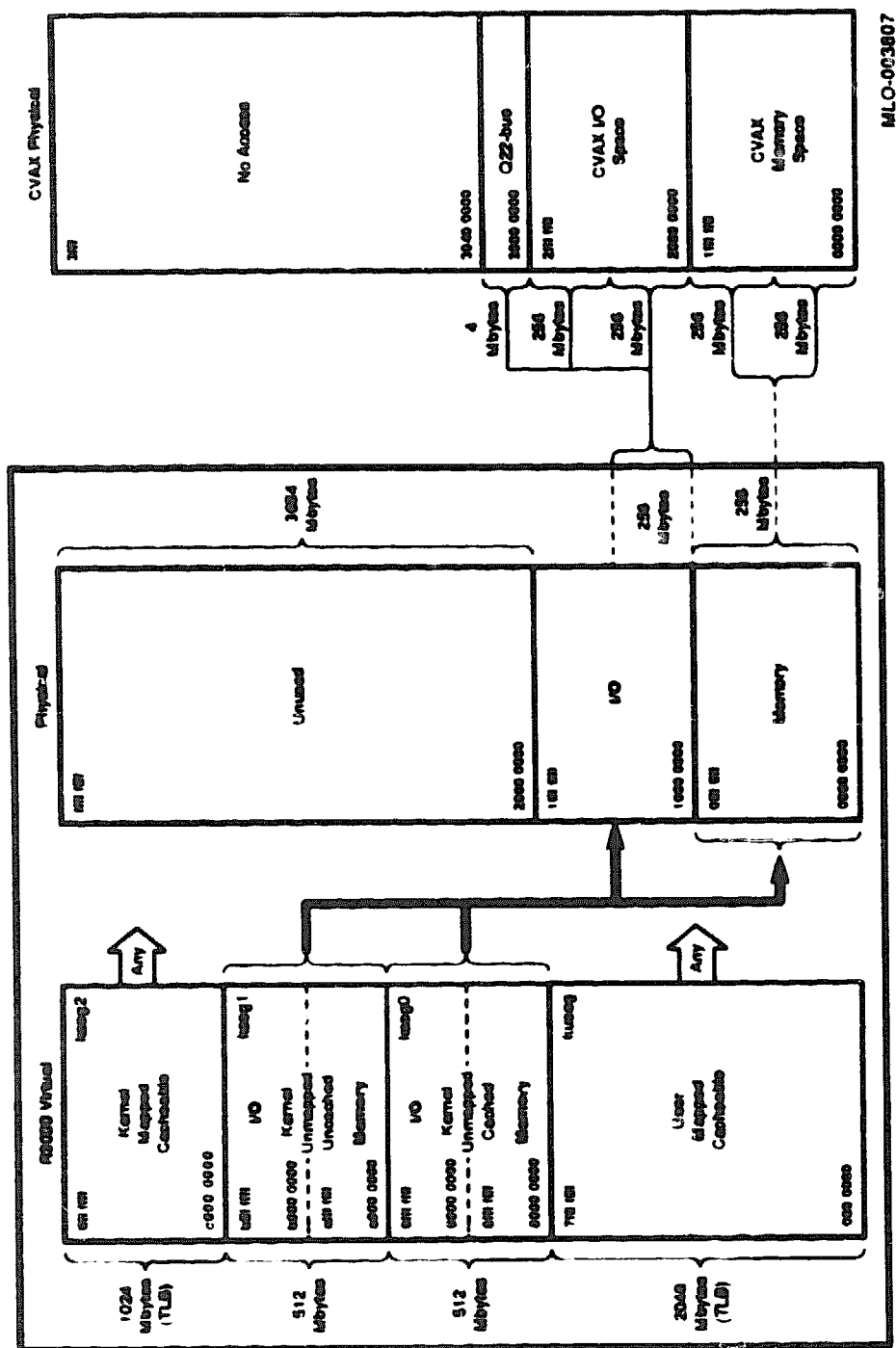
Figure B-1 shows the virtual memory map for the R3000 and CVAX processors. Note that the R3000 virtual addresses are separated into kernel segments (ksecs):

- To address physical locations, use the upper four bits of the kernel segment.
- Always reference kernel segment 1 for I/O addresses, which are unmapped and uncached.
- To calculate the virtual address for kernel segment 0, add 80000000 as an offset to the physical address.
- To calculate the virtual address for kernel segment 1, add a0000000 as an offset to the physical address.

For example, from the R3000 processor:

- If you are using kernel segment 0 (80000000; unmapped and cached), use 8001a340 to access physical location 0001a340.
- If you are using kernel segment 1 (a0000000; unmapped and uncached), use a001a340 to access physical location 0001a340.
- Use a008000c to access DMA error address register 1008000c (physical).

Figure B-1: KN210 Virtual Memory Map



MLO-0003807

## B.2 KN210 CPU Module

Sections B.2.1 through B.2.4 list contents and address ranges for the KN210 CPU module (M7635-AA).

### B.2.1 R3000 Physical Address Space Map (M7635-AA)

**Table B-1: R3000 Memory Space**

<b>Contents</b>	<b>Address Range</b>
Local memory space (up to 64 Mbytes)	0000 0000-03FF FFFF
Reserved memory space	0400 0000-0FFF FFFF

**Table B-2: R3000 Input/Output Space**

<b>Contents</b>	<b>Address Range</b>
Reserved Q22-bus I/O space	1000 0000-1000 0007
Q22-bus floating address space	1000 0008-1000 07FF
User reserved Q22-bus I/O space	1000 0800-1000 0FFF
Reserved and fixed CSR Q22-bus I/O space	1000 1000-1000 1F3F
Interprocessor communication register	1000 1F40
Reserved Q22-bus I/O space	1000 1F42-1000 1FFF
Reserved I/O module address space	1000 2000-1003 FFFF
Two copies of local ROM	1004 0000-1007 FFFF
DMA system configuration register	1008 0000
DMA system error register	1008 0004
Q-bus error address register	1008 0008
DMA error address register	1008 000C
Q22-bus map base register	1008 0010
Reserved	1008 0014-1008 00FF
Main memory configuration register 00	1008 0100
Main memory configuration register 01	1008 0104
Main memory configuration register 02	1008 0108
Main memory configuration register 03	1008 010C
Main memory configuration register 04	1008 0110
Main memory configuration register 05	1008 0114
Main memory configuration register 06	1008 0118
Main memory configuration register 07	1008 011C
Main memory configuration register 08	1008 0120
Main memory configuration register 09	1008 0124
Main memory configuration register 10	1008 0128



**Table B-2 (Cont.): R3000 Input/Output Space**

<b>Contents</b>	<b>Address Range</b>
Main memory configuration register 11	1008 012C
Main memory configuration register 12	1008 0130
Main memory configuration register 13	1008 0134
Main memory configuration register 14	1008 0138
Main memory configuration register 15	1008 013C
Main memory error status register	1008 0140
Main memory control/diagnostics status register	1008 0144
Reserved I/O module address space	1008 0148-1008 3FFF
Interrupt status	1008 4000
Boot and diagnostic register	1008 4004
Select processor register	1008 4008
Reserved I/O module address space	1008 400C-1008 7FFF
Q22-bus map registers	1008 8000-1008 FFFF
Reserved I/O module address space	1009 0000-1013 FFFF
SSC base address register	1014 0000
Reserved	10140004-1014000F
SSC configuration register	1014 0010
Reserved	10140014-1014001F
CDAL bus timeout control register	1014 0020
Reserved	10140024-1014002F
Diagnostic LED register	1014 0030
Reserved I/O module address space	1014 0034-1014 0068
Time-of-year register	1014 006C
Reserved	1014 0070-1014 007C
Console receiver control/status	1014 0080
Console receiver data buffer	1014 0084
Console transmitter control/status	1014 0088
Console transmitter data buffer	1014 008C
Reserved	1014 0090-1014 00DB
I/O system reset register	1014 00DC
Reserved	1014 00E0-1014 00EF
ROM data register	1014 00F0
Bus timeout counter	1014 00F4
Interval timer	1014 00F8
Reserved	1014 00FC-1014 00FF
Timer 0 control register	1014 0100
Timer 0 interval register	1014 0104
Timer 0 next interval register	1014 0108
Timer 0 interrupt vector	1014 010C
Timer 1 control register	1014 0110

**Table B-2 (Cont.): R3000 Input/Output Space**

<b>Contents</b>	<b>Address Range</b>
Timer 1 interval register	1014 0114
Timer 1 next interval register	1014 0118
Timer 1 interrupt vector	1014 011C
Reserved	1014 0120-1014 012F
Address decode match register 0	1014 0130
Reserved	10140138-1014013F
Address decode mask register 0	1014 0134
Address decode match register 1	1014 0140
Address decode mask register 1	1014 0144
Reserved	1014 0148-1014 03FF
Battery backed-up RAM	1014 0400-1014 07FF
Reserved	10140800-13FF FFFF
Local Q22-bus memory space	1400 0000-143F FFFF
Reserved (4 copies local Q22-bus memory)	1440 0000-14FF FFFF
Reserved	1500 0000-1600 004F
Vector read register 0 <sup>1</sup>	1600 0050
Vector read register 1 <sup>1</sup>	1600 0054
Vector read register 2 <sup>1</sup>	1600 0058
Vector read register 3 <sup>1</sup>	1600 005C
Reserved	1600 0060-16FF FFFF
Write-error address register <sup>1</sup>	1700 0000
Reserved	1700 0004-17BF FFFF
ROM <sup>1</sup>	1FC0 0000-1FC1 FFFF
Reserved	1FC2 0000-FFFF FFFF

<sup>1</sup> Accessible only from R3000 processor.

## B.2.2 KN210 Diagnostic Processor Physical Address Space Map (M7635-AA)

**Table B-3: Diagnostic Processor Memory Space**

<b>Contents</b>	<b>Address Range</b>
Local memory space (up to 64 Mbytes)	0000 0000-03FF FFFF
Reserved memory space	0400 0000-1FFF FFFF

**Table B-4: Diagnostic Processor Input/Output Space**

<b>Contents</b>	<b>Address Range</b>
Reserved Q22-bus I/O space	2000 0000-2000 0007
Q22-bus floating address space	2000 0008-2000 07FF
User reserved Q22-bus I/O space	2000 0800-2000 0FFF
Reserved and fixed CSR Q22-bus I/O space	2000 1000-2000 1F3F
Interprocessor communication register	2000 1F40
Reserved Q22-bus I/O space	2000 1F42-2000 1FFF
Reserved I/O module address space	2000 2000-2003 FFFF
MicroVAX system type register (in ROM)	2004 0004
Local ROM; halt mode	2004 0000-2005 FFFF
Local ROM; run mode	2006 0000-2007 FFFF
DMA system configuration register	2008 0000
DMA system error register	2008 0004
Q-bus error address register	2008 0008
DMA error address register	2008 000C
Q22-bit map base register	2008 0010
Reserved	2008 0014-2008 00FF
Main memory configuration register 00	2008 0100
Main memory configuration register 01	2008 0104
Main memory configuration register 02	2008 0108
Main memory configuration register 03	2008 010C
Main memory configuration register 04	2008 0110
Main memory configuration register 04	2008 0110
Main memory configuration register 05	2008 0114
Main memory configuration register 06	2008 0118
Main memory configuration register 07	2008 011C
Main memory configuration register 08	2008 0120
Main memory configuration register 09	2008 0124
Main memory configuration register 10	2008 0128
Main memory configuration register 11	2008 012C

**Table B-4 (Cont.): Diagnostic Processor Input/Output Space**

<b>Contents</b>	<b>Address Range</b>
Main memory configuration register 12	2008 0130
Main memory configuration register 13	2008 0134
Main memory configuration register 14	2008 0138
Main memory configuration register 15	2008 013C
Main memory error status register	2008 0140
Main memory control/diagnostics status register	2008 0144
Reserved I/O module address space	2008 0148–2008 3FFF
Interrupt status	2008 4000
Boot and diagnostic register	2008 4004
Select processor register	2008 4008
Reserved I/O module address space	2008 400C–2008 7FFF
Q22-bus map registers	2008 8000–2008 FFFF
Reserved I/O module address space	2009 0000–2013 FFFF
SSC base address register	2014 0000
Reserved	2014 0004–2014 000F
SSC configuration register	2014 0010
Reserved	2014 0014–2014 001F
CDAL bus timeout control register	2014 0020
Reserved	2014 0024–2014 002F
Diagnostic LED register	2014 0030
Reserved I/O module address space	2014 0034–2014 0068
Time-of-year register	2014 006C
Reserved	2014 0070–2014 007C
Console receiver control/status	2014 0080
Console receiver data buffer	2014 0084
Console transmitter control/status	2014 0088
Console transmitter data buffer	2014 008C
Reserved	2014 0090–2014 00DB
I/O system reset register	2014 00DC
Reserved	2014 00E0–2014 00EF
ROM data register	2014 00F0
Bus timeout counter	2014 00F4
Interval timer	2014 00F8
Reserved	2014 00FC–2014 00FF
Timer 0 control register	2014 0100
Timer 0 interval register	2014 0104
Timer 0 next interval register	2014 0108
Timer 0 interrupt vector	2014 010C
Timer 1 control register	2014 0110
Timer 1 interval register	2014 0114

**Table B-4 (Cont.): Diagnostic Processor Input/Output Space**

<b>Contents</b>	<b>Address Range</b>
Timer 1 next interval register	2014 0118
Timer 1 interrupt vector	2014 011C
Reserved	2014 0120–2014 012F
Address decode match register 0	2014 0130
Address decode mask register 0	2014 0134
Reserved	2014 0138–2014 013F
Address decode match register 1	2014 0140
Address decode mask register 1	2014 0144
Reserved	2014 0148–2014 033F
Battery backed-up RAM	2014 0400–2014 07FF
Reserved I/O module address space	2014 0800–2FFF FFFF
Local Q22-bus memory space	3000 0000–303F FFFF
Reserved I/O module address space	3040 0000–3FFF FFFF

### B.2.3 Diagnostic Processor Registers

Several KN210 internal processor registers (IPRs) are implemented in the SSC chip rather than the CVAX chip. These registers are listed in Table B-5. The R3000 accesses these registers through R3000 memory space.

**Table B-5: Diagnostic Processor Registers**

<b>Dec</b>	<b>Hex</b>	<b>Register Name</b>	<b>Mnemonic</b>	<b>Type</b>	<b>Location</b>
0	0	Kernel Stack Pointer	KSP	r/w	CVAX
1	1	Executive Stack Pointer	ESP	r/w	CVAX
2	2	Supervisor Stack Pointer	SSP	r/w	CVAX
3	3	User Stack Pointer	USP	r/w	CVAX
4	4	Interrupt Stack Pointer	ISP	r/w	CVAX
5	5	Reserved			CVAX
6	6	Reserved			CVAX
7	7	Reserved			CVAX
8	8	P0 Base Register	POB	r/w	CVAX
9	9	P0 Length Register	POLR	r/w	CVAX
10	A	P1 Base Register	P1BR	r/w	CVAX
11	B	P1 Length Register	P1LR	r/w	CVAX
12	C	System Base Register	SBR	r/w	CVAX
13	D	System Length Register	SLR	r/w	CVAX
14	E	Reserved			CVAX

**Table B-5 (Cont.): Diagnostic Processor Registers**

<b>Dec</b>	<b>Hex</b>	<b>Register Name</b>	<b>Mnemonic</b>	<b>Type</b>	<b>Location</b>
15	F	Reserved			CVAX
16	10	Process Control Block Base	PCBB	r/w	CVAX
17	11	System Control Block Base	SCBB	r/w	CVAX
18	12	Interrupt Priority Level	IPL	r/w	CVAX
19	13	AST Level	ASTLVL	r/w	CVAX
20	14	Software Interrupt Request	SIRR	w	CVAX
21	15	Software Interrupt Summary	SISR	r/w	CVAX
22	16	Reserved			CVAX
23	17	Reserved			CVAX
24	18	Interval Clock Control Status	ICCS	r/w	CVAX
25	19	Next Interval Count	NICR	w	CVAX
26	1A	Interval Count	ICR	r	CVAX
27	1B	Time-of-year Register	TOY	r/w	SSC
28	1C	Console Storage Receiver Status	CSRS <sup>1</sup>	r/w	SSC
29	1D	Console Storage Receiver Data	CSRD <sup>1</sup>	r	SSC
30	1E	Console Storage Transmitter Status	CSTS <sup>1</sup>	r/w	SSC
31	1F	Console Storage Transmitter Data	CSDB <sup>1</sup>	w	SSC
32	20	Console Receiver Control Status	RXCS	r/w	SSC
33	21	Console Receiver Data Buffer	RXDB	r	SSC
34	22	Console Transmitter Control Status	TXCS	r/w	SSC
35	23	Console Transmitter Data Buffer	TXDB	w	SSC
36	24	Translation Buffer Disable	TBDR	r/w	CVAX
37	25	Cache Disable	CADR	r/w	CVAX
38	26	Machine Check Error Summary	MCESR	r/w	CVAX
39	27	Memory System Error	MSER	r/w	CVAX
40	28	Reserved			CVAX
41	29	Reserved			CVAX
42	2A	Console Saved PC	SAVPC	r	CVAX
43	2B	Console Saved PSL	SAVPSL	r	CVAX
44	2C	Reserved			CVAX
45	2D	Reserved			CVAX
46	2E	Reserved			CVAX
47	2F	Reserved			CVAX
55	47	I/O System Reset Register	IORESET	—	CVAX

## B.2.4 Global Q22-bus Memory Space Map

**Table B-6: Q22-bus Memory Space**

<b>Contents</b>	<b>Address Range</b>
Q22-bus memory space (octal)	0000 0000–1777 7777

**Table B-7: Q22-bus I/O Space with BBS7 Asserted**

<b>Contents</b>	<b>Address Range</b>
Q22-bus I/O space (octal)	1776 0000–1777 7777
Reserved Q22-bus I/O space	1776 0000–1776 0007
Q22-bus floating address space	1776 0010–1776 3777
User-reserved Q22-bus I/O space	1776 4000–1776 7777
Reserved and fixed CSR Q22-bus I/O space	1777 0000–1777 7477
Interprocessor communication register	1777 7500
Reserved Q22-bus I/O space	1777 7502–1777 7777

## B.3 KN210 I/O Module

Sections B.3.1 and B.3.2 list the contents and address ranges for the KN210 I/O module (M7636-AA).

### B.3.1 R3000 Physical I/O Address Space Map (M7636-AA)

**Table B-8: R3000 Physical Addresses**

<b>Contents</b>	<b>Address Range</b>
Reserved I/O module address space	1000 2000-1003 FFFF
Reserved I/O module address space	1008 0148-1008 3FFF
Reserved I/O module address space	1008 400C-1008 4200
NI station address ROM	1008 4200-1008 427C
Reserved I/O module address space	1008 4280-1008 43FF
NI register data port	1008 4400
NI register address port	1008 4404
Reserved I/O module address space	1008 4408-1008 45FF
MSI diagnostic register 0	1008 4600
MSI diagnostic register 1	1008 4604
MSI diagnostic register 2	1008 4608
MSI control and status register	1008 460C
MSI ID register	1008 4610
Reserved MSI register	1008 4614
Reserved MSI register	1008 4618
MSI timeout register	1008 461C
Reserved MSI register	1008 4620
Reserved MSI register	1008 4624
Reserved MSI register	1008 4628
Reserved MSI register	1008 462C
Reserved MSI register	1008 4630
Reserved MSI register	1008 4634
MSI short target list pointer	1008 4638
MSI long target list pointer	1008 463C
MSI initiator list pointer	1008 4640
MSI DSSI control register	1008 4644
MSI DSSI status register	1008 4648
Reserved MSI register	1008 464C
Reserved MSI register	1008 4650
MSI diagnostic control register	1008 4654
MSI clock control register	1008 4658
MSI internal state register 0	1008 465C



**Table B-8 (Cont.): R3000 Physical Addresses**

<b>Contents</b>	<b>Address Range</b>
MSI internal state register 1	1008 4660
MSI internal state register 2	1008 4664
MSI internal state register 2	1008 4668
Reserved MSI register	1008 466C
Reserved MSI register	1008 4670
Reserved MSI register	1008 4674
Reserved MSI register	1008 4678
Reserved MSI register	1008 467C
Reserved I/O module address space	1008 4680-1008 7FFF
Reserved I/O module address space	1009 0000-1009 FFFF
MSI buffer RAM	1010 0000-1011 FFFF
NI buffer RAM	1012 0000-1013 FFFF

### **B.3.2 Diagnostic Physical I/O Address Space Map (M7636-AA)**

**Table B-9: Diagnostic Input/Output Addresses**

<b>Contents</b>	<b>Address Range</b>
Reserved I/O module address space	2000 2000-2003 FFFF
Reserved I/O module address space	2008 0148-2008 3FFF
Reserved I/O module address space	2008 400C-2008 4200
NI station address ROM	2008 4200-2008 427C
Reserved I/O module address space	2008 4280-2008 43FF
NI register data port	2008 4400
NI register address port	2008 4404
Reserved I/O module address space	2008 4408-2008 45FF
MSI diagnostic register 0	2008 4600
MSI diagnostic register 1	2008 4604
MSI diagnostic register 2	2008 4608
MSI control and status register	2008 460C
MSI ID register	2008 4610
Reserved MSI register	2008 4614
Reserved MSI register	2008 4618
MSI timeout register	2008 461C
Reserved MSI register	2008 4620
Reserved MSI register	2008 4624
Reserved MSI register	2008 4628

**Table B-9 (Cont.): Diagnostic Input/Output Addresses**

<b>Contents</b>	<b>Address Range</b>
Reserved MSI register	2008 462C
Reserved MSI register	2008 4630
Reserved MSI register	2008 4634
MSI short target list pointer	1008 4638
MSI long target list pointer	2008 463C
MSI initiator list pointer	2008 4640
MSI DSSI control register	2008 4644
MSI DSSI status register	2008 4648
Reserved MSI register	2008 464C
Reserved MSI register	2008 4650
MSI diagnostic control register	2008 4654
MSI clock control register	2008 4658
MSI internal state register 0	2008 465C
MSI internal state register 1	2008 4660
MSI internal state register 2	2008 4664
MSI internal state register 2	2008 4668
Reserved MSI register	2008 466C
Reserved MSI register	2008 4670
Reserved MSI register	2008 4674
Reserved MSI register	2008 4678
Reserved MSI register	2008 467C
Reserved I/O module address space	2008 4680-2008 7FFF
Reserved I/O module address space	2009 0000-2009 FFFF
MSI buffer RAM	2010 0000-2011 FFFF
NI buffer RAM	2012 0000-2013 FFFF



## Appendix C

# Configuring the KFQSA

---

**This appendix describes the KFQSA storage adapter and explains how to:**

- **Configure the KFQSA storage adapter at installation**
- **Enter console I/O mode**
- **Run the Configure utility**
- **Program the EEROM on the KFQSA**
- **Reprogram the EEROM on the KFQSA**
- **Change the ISE's allocation class and unit number**

### C.1 KFQSA Overview

The KFQSA module is a storage adapter that allows Q-bus host systems that support the KFQSA to communicate with storage peripherals based on the Digital Storage Architecture (DSA), using the Digital Storage System Interconnect (DSSI). In a KN210-based system, one KFQSA module can connect up to six RF-series integrated storage elements (ISEs) to the host system, using a single DSSI bus cable.

The KFQSA contains the addressing logic required to make a connection between the host and a requested ISE on the DSSI bus. Each ISE has its own controller, which contains the intelligence and logic necessary to control data transfers over the DSSI bus. The KFQSA presents a mass storage control protocol (MSCP) U/Q port for each ISE.

The EEROM on the KFQSA contains a configuration table. After you install the KFQSA, you program the EEROM with the CSR address for each ISE in the system.

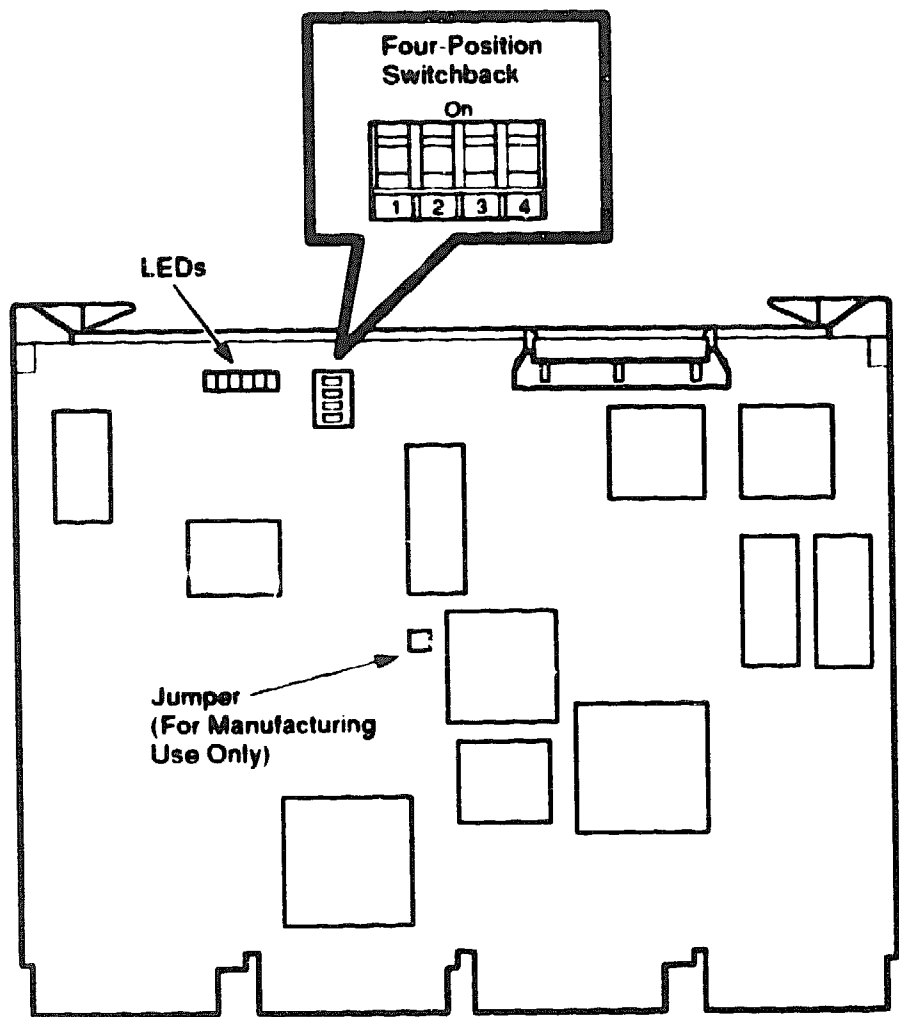
## C.2 Configuring the KFQSA at Installation

At installation, configure the KFQSA as follows:

**CAUTION:** *Static electricity can damage integrated circuits. Use the wrist strap and antistatic mat found in the Antistatic Kit (29-26246) when you work with the internal parts of a computer system.*

1. Check the KFQSA module for the presence of a jumper intended for manufacturing use only. The location of this jumper is shown in Figure C-1. Remove the jumper, if present.
2. To set a temporary CSR address that enables you to access the EEROM, use the four-position DIP switchpack shown in Figure C-1 as follows:
  - a. Set switches 1, 2, 3, and 4 to reflect a fixed CSR address to allow the KFQSA to be programmed. Example C-1 shows the correct switch settings for a system with only one KFQSA.
  - b. Install the KFQSA adapter module into the backplane according to the procedures in the appropriate enclosure maintenance manual.

**Figure C-1: KFQSA Module Layout (M7769)**



MLO-001878

## Example C-1: KFQSA (M7769) Service Mode Switch Settings

### KFQSA Four-Position Switchpack

Switches:	S/N Mode 1	Fx/F1 2	MSB 3	LSB 4	Fixed Address
First KFQSA	on	off	on	on	0774420
Additional:	on	off	on	off	0774424
	on	off	off	on	0774430
	on	off	off	off	0774434

S/N = Service mode/Normal operating mode

Fx/F1 = fixed/floating CSR address

### C.2.1 Entering Maintenance Mode

After installing the KFQSA, you issue a series of commands to the KN210 system at the maintenance mode prompt (>>>) in order to program the EEROM on the KFQSA. You may type these commands in either uppercase or lowercase letters. Unless otherwise specified, type each command, then press **Return**.

Enter maintenance mode as follows:

1. Set the operation switch on the H3602-AB CPU cover panel to the maintenance position (⊕).
2. Set the function switch on the CPU cover panel to the enable position (⊙).
3. Set the on/off power switch to on (1).
4. When the power-up self-tests complete, the console prompt appears, as shown in Example C-2.

## Example C-2: Entering Console Mode Display

Performing normal system tests.

51..50..49..48..47..46..45..44..43..42..41..40..39..38..37..36..35..  
34..33..32..31..30..29..28..27..26..25..24..23..22..21..20..19..18..  
17..16..15..14..13..12..11..10..09..08..07..06..05..04..03..

Tests completed.

>>>

## C.2.2 Displaying Current Addresses

Type **SHOW QBUS** to display the current Q22-bus addresses (Example C-3).  
Note that the KFQSA adapter appears in service mode as KFQSA #0.

### Example C-3: SHOW QBUS Display

>>> SHOW QBUS

Scan of Qbus I/O Space

-20001910 (774420) = 0000 (000) KFQSA #0  
-20001912 (774422) = 0AA0  
-20001920 (774440) = FF08 (120) DELQA/DEQNA/DESQA  
-20001922 (774442) = FF00  
-20001924 (774444) = FF2B  
-20001926 (774446) = FF09  
-20001928 (774450) = FFA3  
-2000192A (774452) = FF96  
-2000192C (774454) = 8000  
-2000192E (774456) = 1030  
-20001940 (774500) = 0000 (260) TQK50/TQK70/TU81E/RV20/KFQSA-TAPE  
-20001942 (774502) = 0BC0  
-20001F40 (777500) = 0020 (004) IPCR

Scan of Qbus Memory Space

>>>



### C.2.3 Running the Configure Utility

Now that you have physically reconfigured the system by installing the KFQSA storage adapter, you must run the Configure utility to find the correct address for each device and module in the system. The Configure utility uses Q-bus/UNIBUS fixed and floating address space rules.

Run the Configure utility as follows. Refer to Example C-4.

1. At the maintenance mode prompt, type **CONFIGURE**, then type **HELP** at the **Device, Number?** prompt for a list of devices that can be configured.

**NOTE:** *Some of the devices listed in the **HELP** display are not supported by the KN210-A CPU.*

2. For each device in the system, type the device name at the **Device, Number?** prompt. If you have more than one of the same type, type a comma followed by the total number of that device. In Example C-4, the system contains one KFQSA with six ISEs.

Be sure you list *all* the devices: those already installed and those you plan to install.

3. Type **EXIT**. The Configure utility displays an address and vector assignment for each device. Example C-4 shows the address and vector assignments and the device input. Write down the addresses for the KFQSA devices.
4. For all modules except the KFQSA, verify that the CSR addresses are set correctly by comparing the addresses listed in the **SHOW QBUS** command with those listed in the Configure utility display. If necessary, remove modules from the backplane and reset switches or jumpers to the addresses in your Configure display, using module removal and replacement procedures in *BA213 Enclosure Maintenance*.

## Example C-4: Configure Display

>>> CONFIGURE

Enter device configuration, HELP, or EXIT

Device, Number? help

Devices:

LPV11	RXJ11	DLV11J	DZQ11	DZV11	DFA01
RLV12	TSV05	RXV21	DRV11W	DRV11B	DPV11
DMV11	DELQA	DEQNA	DESQA	RQDX3	KDA50
RRD50	RQC25	KFQSA-DISK	TQK50	TQK70	TU61E
RV20	KFQSA-TAPE	KMV11	IEQ11	DHQ11	DHV11
CKA16	CXB16	CXY08	VCB01	QVSS	LVN11
LVN21	QPSS	DSV11	ADV11C	AAV11C	AXV11C
KWV11C	ADV11D	AAV11D	VCB02	QDSS	DRV11J
DRQ3B	VSV21	IBQ01	IDV11A	IDV11B	IDV11C
IDV11D	IAV11A	IAV11B	MIRA	ADQ32	DTC04
DESNA	IGQ11				

Numbers:

1 to 255, default is 1

Device, Number? KFQSA-DISK, 6

Device, Number? DESQA

Device, Number? TQK70

Device, Number? EXIT

Address/Vector Assignments

-774440/120 DESQA

-772150/154 KFQSA-DISK !Node 0 (assigned in order, 0 to n)

-760334/300 KFQSA-DISK !Node 1

-760340/304 KFQSA-DISK !Node 2

-760344/310 KFQSA-DISK !Node 3

-760350/314 KFQSA-DISK !Node 4

-760354/320 KFQSA-DISK !Node 5

-774500/260 TQK70

## **C.3 Programming the KFQSA**

**Program the configuration table in the EEROM of the KFQSA to include all ISEs on the DSSI bus, as follows. Refer to Examples C-5, C-6, and C-7.**

- 1. Determine the DSSI node plug address for each ISE you are configuring. Start with node 0, then continue up through node 5. In Example C-4, nodes 0, 1, 2, 3, 4, and 5 are used; node 7 is saved for the KFQSA adapter.**
- 2. To program the KFQSA, type SET HOST/UQSSP/MAINT/SERV 0 at the console prompt on each system.**
- 3. Type HELP to display a list of supported commands.**
- 4. Program the KFQSA to include each DSSI device in the system:**
  - a. For each ISE: type SET, followed by the node number, the CSR address (from the list of addresses you obtained from the Configure utility), and the model number (disk ISE's are model 21).**
  - b. Type SHOW to display the configuration table you just programmed.**
  - c. Check the display to make sure the addresses are correct.**
  - d. Type EXIT to save the configuration table or QUIT to delete the table.**

## Example C-5: Display for Programming the First KFQSA

```
>>> SET HOST/UQSSP/MAINT/SERV 0      !0 refers to first KFQSA
                                       !in the system.

UQSSP Controller (772150)

Enter SET, CLEAR, SHOW, HELP, EXIT, or QUIT

Node      CSR Address      Model
  7        ----- KFQSA -----
? HELP
Commands:
    SET <node> /KFQSA                !Sets KFQSA DSSI node
                                       !number
    SET <node> <CSR_address> <model> !Enables a DSSI device
    CLEAR <node>                     !Disables a DSSI device
    SHOW                             !Displays current
                                       !configuration
    HELP                             !Displays this display
    EXIT                             !Saves the KFQSA program
    QUIT                             !Does not save the KFQSA
                                       !program

Parameters:
    <node>                           !0 through 7
    <CSR_address>                     !760010 to 777774
    <model>                           !21 (disk) or 22 (tape)

? SET 0 772150 21
? SET 1 760334 21
? SET 2 760340 21
? SET 3 760344 21
? SET 4 760350 21
? SET 5 760354 21
? SHOW
Node      CSR Address      Model
  0        762105          21
  1        760334          21
  2        760340          21
  3        760344          21
  4        760350          21
  5        760354          21
  7        ----- KFQSA -----
? exit
Programming the KFQSA...              !Note from the system that
                                       !the KFQSA is
                                       !being programmed.
```

5. Turn the system power off by setting the on/off switch to off (0).
6. Remove the KFQSA from the backplane.
7. Confirm that the unit ID plugs on the enclosure's operator control panel (OCP) match the node IDs you just programmed.
8. On the KFQSA, set switch 1 on the four-position switchpack to Off (1). (Figure C-1 shows the location and position of the switchpack.) This action sets the KFQSA to the normal operating mode; switches 2, 3, and 4 are ignored since switch 1 is set to off, and the DSSI addresses are read from the EEROM.
9. Reinstall the KFQSA in the backplane.
10. Power on the system by setting the on/off switch to on (1). Wait for the self-tests to complete.
11. At the maintenance mode prompt, type `SHOW QBUS` to verify that all addresses are present and correct, as shown in Example C-6.
12. Type `SHOW DEVICE` to verify that all ISEs are displayed correctly, as shown in Example C-7.

## Example C-6: SHOW QBUS Display

>>> SHOW QBUS

Scan of Qbus I/O Space

```
-200000DC (760334)=0000 (300) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000DE (760336)=0AA0
-200000E0 (760340)=0000 (304) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000E2 (760342)=0AA0
-200000E4 (760344)=0000 (310) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000E6 (760346)=0AA0
-200000E8 (760350)=0000 (314) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000EA (760352)=0AA0
-200000EC (760354)=0000 (320) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-200000EE (760356)=0AA0
-20001468 (772150)=0000 (154) RQDX3/KDA50/RRD50/RQC25/KFQSA-DISK
-2000146A (772152)=0AA0
-20001920 (774440)=FF08 (120) DELQA/DEQNA/DESQA
-20001922 (774442)=FF00
-20001924 (774444)=FF2B
-20001926 (774446)=FF09
-20001928 (774450)=FFA3
-2000192A (774452)=FF96
-2000192C (774454)=0050
-2000192E (774456)=1030
-20001940 (774500)=0000 (260) TQK50/TQK70/TU81E/RV20/KFQSA-TAPE
-20001942 (774502)=0BC0
-20001F40 (775000)=0040 IPCR
```

Scan of Qbus Memory Space

>>>

## Example C-7: SHOW DEVICE Display

>>> SHOW DEVICE

DSSI Node 0 (772150)

-DIA0 -rf(0,0,\*) (RF71)

DSSI Node 1 (760334)

-DIA1 -rf(1,1,\*) (RF71)

DSSI Node 2 (760340)

-DIA2 -rf(2,2,\*) (RF71)

DSSI Node 3 (760344)

-DIA3 -rf(3,3,\*) (RF71)

DSSI Node 4 (760350)

-DIA4 -rf(4,4,\*) (RF71)

DSSI Node 5 (760354)

-DIA5 -rf(5,5,\*) (RF71)

DSSI Node 7 (\*)

QSSP Tape Controller 0 (774500)

-MUA0 -tm(0,0) (TK70)

Ethernet Adapter 0 (774440)

-ESA0 -se -mop() (08-00-2B-0C-C4-75)

## **C.4 Reprogramming the KFQSA**

When you add a new DSSI device to a system with at least one RF-series ISE that has been programmed correctly, you must reprogram each KFQSA on the DSSI bus to include the new device(s) as follows:

1. Enter maintenance mode, using the procedure in Section C.2.1.
2. At the maintenance mode prompt, type `SHOW DEVICE` for a display of all devices currently in the system. The display includes tape drives and the Ethernet adapter, as shown in Section C.3, Example C-7.

This display lists the Q-bus address and port name of the device, such as DUA0 RF71.

3. Type `SHOW QBUS` for a display of the eight-digit Q-bus address (hex) for each device, as shown in Section C.3, Example C-6.
4. Find the eight-digit Q-bus address for an ISE attached to the KFQSA that you are reprogramming. Use the `SET HOST` command to enter the KFQSA through an existing port and edit the configuration table as follows. Refer to Example C-8.
  - a. Type `SET HOST/UQSSP/MAINT` followed by the Q-bus address.
  - b. Use the `SET` and `CLEAR` commands to reconfigure the KFQSA, as shown in Example C-8.
  - c. Type `SHOW` to display the new KFQSA configuration table setting.
  - d. Type `EXIT` to save the configuration table or `QUIT` to cancel the reprogramming.



## Example C-8: Reprogramming the KFQSA Display

```
>>> SET HOST/UQSSP/MAINT 20001468
UQSSP Controller (772150)
```

Node	CSR Address	Model
0	772150	21
1	760334	21
2	760340	21
3	760344	21
4	760350	21
5	760354	21
7	----- KFQSA -----	

```
? CLEAR 5
```

```
? SHOW
```

Node	CSR Address	Model
0	772150	21
1	760334	21
2	760340	21
3	760344	21
4	760350	21
7	----- KFQSA -----	

```
? SET 5 760354 21
```

```
? SHOW
```

Node	CSR Address	Model
0	772150	21
1	760334	21
2	760340	21
3	760344	21
4	760350	21
5	760354	21
7	----- KFQSA -----	

```
? EXIT
```

```
Programming the KFQSA...
```

```
!Note from the system that the
!KFQSA is being programmed.
```

## C.5 Changing the ISE Unit Number

This section describes how to change the ISE unit number. For most configurations, you will not need to change the default unit numbers.

**NOTE:** *If you add a new device to the system, you must rebuild the ULTRIX-32 kernel to include the device.*

Change the unit number, using the console-based DUP driver utility, as follows. Refer to Example C-9.

1. Enter maintenance mode, using the procedure in Section C.2.1.
2. At the maintenance mode prompt, type SET HOST/DUP/UQSSP/DISK 0 PARAMS (0 through 6 for the ISE to which you want to connect) to start the DUP server.
3. Type SHOW UNITNUM to check the unit number.
4. To change the ISE's unit number from the default value, type SET UNITNUM *n* (where *n* is the new unit number). For example, type SET UNITNUM 20 to change the unit number from 0 to 20.
5. Type SET FORCEUNI 0 to set the forceunit flag to zero in order to use a nondefault value. If you do not change the FORCEUNI parameter, the drive unit number defaults to the number of the corresponding DSSI plug on the operator control panel (OCP).
6. Type SHOW UNITNUM to show the new unit number.
7. Type SHOW FORCEUNI to show the new forceunit flag values.
8. Type WRITE, then type Y to save the new values into the EEROM or N to cancel the reprogramming.
9. Type SHOW DEVICE to make sure you have programmed the first ISE to have a unit number of 20. When you boot the operating system, the display shows the new unit number.

## Example C-9: Display for Changing Unit Number

>>> SET HOST/DUP/UQSSP/DISK 0 PARAMS

Starting DUP server...

UQSSP Disk Controller 0 (772150)

Copyright (c) 1988 Digital Equipment Corporation

PARAMS> SHOW UNITNUM

Parameter	Current	Default	Type	Radix
UNITNUM	0	0	Word	Dec B

PARAMS> SET UNITNUM 20

PARAMS> SET FORCEUNI 0

PARAMS> SHOW UNITNUM

Parameter	Current	Default	Type	Radix
UNITNUM	20	0	Word	Dec U

PARAMS> SHOW FORCEUNI

Parameter	Current	Default	Type	Radix
FORCEUNI	0	1	Boolean	0/1 U

PARAMS> WRITE

Stopping DUP server...

>>> SHOW DEVICE

DSSI Node 0 (772150)

-DIA20 -rf(0,20,\*) (RF71)

DSSI Node 1 (760334)

-DIA1 -rf(1,1,\*) (RF71)

DSSI Node 2 (760340)

-DIA2 -rf(2,2,\*) (RF71)

DSSI Node 3 (760344)

-DIA3 -rf(3,3,\*) (RF71)

DSSI Node 4 (760350)

-DIA4 -rf(4,4,\*) (RF71)

DSSI Node 5 (760354)

-DIA5 -rf(5,5,\*) (RF71)

DSSI Node 7 (\*)

Example C-9 Cont'd on next page

### **Example C-9 (Cont.): Display for Changing Unit Number**

UQSSP Tape Controller 0 (774500)  
-MUA0 -tm(0,0) (TK73)

Ethernet Adapter 0 (774440)  
-ESA0 -se -mop() (08-00-2E-0C-C4-75)

[illegible]

APPENDIX D

[illegible]

## Appendix D

# Field Replaceable Units (FRUs)

This appendix lists the major field replaceable units (FRUs) for the KN210-based system (DECsystem 5400).

FRU Description	Part Number
<b>KN210 Base System</b>	
H3602-AB bulkhead assembly	70-25775-02
KN210 CPU module	M7635-AA
KN210 I/O module	M7636-AA
MS650-BA memory module, 16 Mbyte	M7622-AA
<b>Digital Storage System Interconnect (DSSI)</b>	
DSSI cable, external	17-02152-03
DSSI cable, round	17-02059-01
DSSI daisy-chain cable, flat	17-01836-01
DSSI operator control panel (OCP)	70-25752-01
DSSI terminator	12-29258-01
KFQSA module	M7769-00
RF30/71 drive bracket	70-36498-01
RF71 drive module	54-18316-01
RF71 head disk assembly (HDA)	70-23557-01
RF71 integrated storage element (ISE)	RF71-EA
RF71 shock mount, bottom	70-25452-03
RF71 shock assembly, top	70-25452-04
RF-series lens, encoder set	12-28766-19

FRU Description	Part Number
<b>Digital Storage Architecture (DSA)</b>	
RA70 electronic control module (ECM)	70-22494-01
RA70 head disk assembly (HDA)	70-21946-01
RA70 operator control panel (OCP)	54-17232-02
RA70/KDA50 signal cable	17-00251-03
RA90 electronic control module (ECM)	70-22942-02
RA90 head disk assembly (HDA)	70-22951-01
RA-series drive bracket, top	70-24559-01
RA-series drive bracket, bottom	70-24559-02
RA-series drive cable, 4-pin	17-01503-01
RA-series drive cable, 20-pin	17-00847-06
RA-series drive shock mount 1	70-23997-05
RA-series drive shock mount 2	70-23997-06
SDI cable, external	17-00464-12
<b>Cables, Base System</b>	
Backplane to operator control panel (OCP) cable	17-01964-01
KN210 CPU module to KN210 I/O module	17-02418-01
Memory cable	17-01898-01
Memory cable	17-01898-02
Memory cable	17-01898-03
TK-series tape drive power cable	17-01937-01
H3602-AB cable	17-02430-01
<b>Option Modules</b>	
CXA16-M module	M3118-YA
CXB16-M module	M3118-YB
CKY08-M module	M3119-YA
DEQNA-SA module	M7504-PA
DELQA-SA module	M7516-PA
DESQA-SA module	M3127-PA
DPV11-SA module	M8086-PA
KDA50 modules	M7164-00
	M7165-00
KLESI-SA controller module	M7740-PA
Load module	M9060-YA
LPV11-SA module	M8086-PA
TQK70 controller module	M7559-00

FRU Description	Part Number
<b>BA213 Enclosure</b>	
Battery pack	12-19245-01
Blank labels	36-26883-01
Backplane, split bus DC	70-23712-01
Bulkhead cover, single	70-23981-01
Bulkhead cover, dual	70-23982-02
CD filler panel	74-33507-01
Fan assembly	70-23988-01
Fan, +12 Vdc	12-23609-04
FCC clip/handles	12-26340-01
Filter input assembly	70-23769-01
EOS clip	12-26922-01
Mass storage power harness, left	17-01989-01
Mass storage power harness, right	17-01990-01
One quarter-turn fastener/handle	12-26948-01
Power supply, 120 Vac	H7868-A
Power supply, 240 Vac	H7868-B
Side gap filler panel (2)	70-24505-01
Switch assembly	70-23999-01
<b>Miscellaneous</b>	
BA213 chassis assembly, 120 Vac	70-24227-03
BA213 chassis assembly, 240 Vac	70-24227-04
Control power switch 886-A	30-30166-01
Control power switch 886-B	30-30166-02
Power controller, 120 Vac	00-874-D
Power controller, 240 Vac	00-874-F
Power control cable, 3-pin	70-06288-06
<b>Loopback Connectors</b>	
CKY06 loopback	12-26964-01
DEQNA Ethernet loopback	12-22196-02
H3103-00 (MMJ) loopback	12-25083-01
H3197-00 CKY06 loopback	12-15336-07





## Appendix E

# Related Documentation

---

The following documents contain information relating to the KN210 CPU module set:

Document Title	Order Number
<b>Modules</b>	
CKA16 Technical Manual	EK-CAB16-TM
CXY08 Technical Manual	EK-CXY08-TM
DELQA Technical Manual	EK-DELQA-UG
DESQA Technical Manual	EK-DESQA-TM
DSV11-S Communications Option User Guide	EK-DSV11-UG
DSV11 Communications Option Technical Description	EK-DSV11-TD
KDA50-Q CPU Module User's Guide	EK-KDA5Q-UG
KFQSA Installation Guide	EK-KFQSA-IN
KN210 CPU Module Set Technical Manual	EK-KN210-TM
<b>Disk and Tape Drives</b>	
RA60 Disk Drive Service Manual	EK-ORA60-SV
RA60 Disk Drive User's Guide	EK-ORA60-UG
RA81 Disk Drive Service Manual	EK-ORA81-SV
RA81 Disk Drive User's Guide	EK-ORA81-UG
RF71 Disk Drive User's Guide	EK-RF71D-UG
<b>Systems</b>	
BA213 Enclosure Maintenance	EK-189AA-MG
B215F Expander Installation	EK-310AA-IN
H9644 Cabinet Maintenance	EK-221AA-MG
R215F Expander Installation	EK-310AA-IN
Microsystems Options	EK-192AA-MG
Microsystems Site Preparation Guide	EK-067AB-PG

<b>Document Title</b>	<b>Order Number</b>
<b>Diagnostics</b>	
MicroVAX Diagnostic Monitor Ethernet Server User's Guide	AA-FNTAC-DN
MicroVAX Diagnostic Monitor Reference Card	AV-FMXAA-DN
MicroVAX Diagnostic Monitor User's Guide	AA-FM7AB-DN
ULTRIX-32 Guide to System Exercisers	AA-ME96A-TE
ULTRIX-32 Guide to the Error Logger System	AA-ME965-TE
<b>Networks</b>	
Ethernet Transceiver Tester User's Manual	EK-ETHTT-UG



# Index

---

! (comment command), 3-78

9E utility, 4-11

    examples, 4-12

9C utility, 4-32, 4-41

## A

---

Acceptance testing, 4-30

Address assignments, B-3 to B-13

## B

---

Boot and diagnostic facility, on  
    KN210 CPU, 1-9

BOOT command, in maintenance  
    mode, 3-45

boot command, in normal mode,  
    3-20

Bootstrap

    of MDM, description of, 3-10

    of MDM, supported boot devices,  
        3-12

    of MDM, supported boot flags,  
        3-12

    of ULTRIX-32, description of,  
        3-9

    of ULTRIX-32, procedures for,  
        3-9

    of ULTRIX-32, supported devices,  
        3-9

    support, for KN210 systems, 3-8

Bus length (DSSI), 2-11

## C

---

Cabling

    DSSI, 2-8

    RF71, 2-8

Cache memory, on KN210 CPU, 1-8

CLANCE chip, on KN210 I/O  
    module, 1-11

CMCTL chip, on KN210 CPU, 1-9

? command, in normal mode, 3-31

Comment command (!), 3-78

Configuration

    and module order, 2-1

    DSSI, 2-4

    rules, 2-2

    worksheet, 2-11

CONFIGURE command, 2-2, 3-46

Console commands

    address space control qualifiers,  
        in maintenance mode, 3-42

    address specifiers, in maintenance  
        mode, 3-37

    binary load and unload (X), 3-76

    BOOT, in maintenance mode,  
        3-45

    boot, in normal mode, 3-20

    ! (comment), 3-78

    CONFIGURE, in maintenance  
        mode, 3-46

    CONTINUE, in maintenance  
        mode, 3-48

    continue, in normal mode, 3-23

    d (deposit), in normal mode, 3-24

    data control qualifiers, 3-41

    DEPOSIT, in maintenance mode,  
        3-49

    dump, in normal mode, 3-25

    e (examine), in normal mode,  
        3-27

    EXAMINE, 3-50

    EXIT, 3-52

    fill, in normal mode, 3-28

    FIND, 3-53

## Console commands (Cont.)

- go, in normal mode, 3-29
- HALT, 3-54
- HELP, 3-55
- help, in normal mode, 3-30
- init, in normal mode, 3-32
- INITIALIZE, 3-57
- ?, in normal mode, 3-31
- keywords, 3-43
- list of, 3-43
- MOVE, 3-59
- NEXT, 3-61
- printenv, in normal mode, 3-33
- qualifier and argument
  - conventions, in maintenance mode, 3-37
- qualifiers, 3-41
- REPEAT, 3-63
- SEARCH, 3-64
- SET, 3-66
- setenv, in normal mode, 3-34
- SHOW, 3-69
- START, 3-73
- symbolic addresses, 3-38
- syntax, in maintenance mode, 3-37
- TEST, 3-74
- UNJAM, 3-75
- unsetenv, in normal mode, 3-35
- X (binary load and unload), 3-76

Console displays, 4-15

- and FRUs, 4-19

Console error messages, 4-27

- list of, 4-28
- sample of, 4-16

Console port, testing, 4-45

Console serial line on KN210 CPU, 1-9

CONTINUE command, 3-48

continue command, in normal mode, 3-23

CQBIC chip, on KN210 CPU, 1-10

Current and power values, 2-12

CVAX

- halt entry and dispatch code, 3-3

## CVAX (Cont.)

- on KN210 CPU, 1-10

CVAX ROM-based diagnostics

- description of, 4-3
- list of, 4-4
- parameters for, 4-4
- utilities, 4-4

## D

d (deposit) command, in normal mode, 3-24

DEPOSIT command, 3-49

Diagnostic executive, 4-3

- error field, 4-17

Diagnostics

- CVAX, description of, 1-10
- KN210 CVAX and boot and diagnostic facility, on KN210 CPU, 1-9

Diagnostic tests

- list of, 4-4
- parameters for, 4-4

Differences, between KN210 and other systems

- in autoboot capability, 1-14
- in console programs, 3-8
- in firmware, 3-1
- in H3602-AB switch meanings, 1-13
- in modules and slot locations, 1-5
- in terminology surrounding word size, 1-8

Displays

- console banner, 3-5

Documentation, relating to KN210 systems, list of, E-1

DRVEXR local program, 4-36, 4-49

DRVST local program, 4-36, 4-48

DSSI

- bus length, 2-11
- bus termination, 2-11
- cabling, 2-8
- configuration, 2-4
- interface, on KN210 I/O module, 1-11

## DSSI (Cont.)

- ISE order, 2-4
- node ID, 2-4
- node name, changing, 2-5
- testing with H3281 loopback, 4-44
- unique addresses, 4-35
- unit number, changing, 2-6
- dump command, in normal mode, 3-25

## E

- e (examine) command, in normal mode, 3-27
- Environment variables, 3-16
- ERASE local program, 4-51
- Error messages
  - console, list of, 4-28
  - console, sample of, 4-16
  - halt, 4-27
  - VMB, 4-29
- Ethernet
  - See Network interface
- Ethernet connectors
  - location of, on H3602-AB, 1-12
- EXAMINE command, 3-50
- EXIT command, 3-52

## F

- FE utility, 4-37
- Field replaceable units (FRUs) for DEC system 5400, D-1 to D-3
- fill command, in normal mode, 3-28
- FIND command, 3-53
- Firmware
  - description of, on KN210 CPU, 3-1
  - features of, 3-2
  - power-up sequence, 3-5
- Firmware, on KN210 CPU, 1-9
- Floating-point accelerator, on KN210 CPU, 1-8
- FRUs and console display, 4-19
- Function switch

## Function switch (Cont.)

- and loopback tests at power-up, 3-6
- and query position at power-up, 3-7
- Function switch, on H3602-AB, 1-13

## G

### General purpose registers (GPRs)

- in error display, 4-19
- initialization of, 3-11
- symbolic addresses for, 3-38
- go command, in normal mode, 3-29

## H

- H3103 loopback connector, 4-45
- H3281 loopback connector for DSSI, 4-44
- H3602-AB CPU I/O panel
  - description of, 1-12
- H3602-AB I/O panel, 4-45
- H8572 loopback connector, 4-45
- HALT command, 3-54
- Halts
  - and actions taken, 3-3
  - conditions for, external, 3-4
  - CVAX, halt entry and dispatch code, 3-3
  - messages, list of, 4-27
  - registers saved, 3-3
  - registers set to fixed values, 3-3
- Hardware error summary register, 4-38
- HELP command, 3-55
- help command, in normal mode, 3-30
- HISTORY local program, 4-36, 4-50

## I

- init command, in normal mode, 3-52
- INITIALIZE command, 3-57
- Initial power-up test
  - See IPT

## Installation

- of KN210 and MS650-BA modules, 1-5

## Internal processor registers (IPRs)

- symbolic addresses for, 3-38

- IPT, 3-5, 4-20

## K

---

### KFQSA storage adapter

- and running Configure utility, C-6
- changing the ISE unit number, C-15
- configuring, C-2
- programming, C-8
- reprogramming, C-13

### KN210

- features of, 1-7

### KN210 CPU module

- description of, 1-1
- firmware, features of, 3-2

### KN210 I/O module

- description of, 1-3

## L

---

### Load module, M9060-YA, 2-11

### Loopback connectors

- H3103, 4-45
- H8572, 4-45
- list of, 4-46
- tests, 4-44

### Loopback tests

- at power-up, 3-6
- for DSSI problems, 4-44
- for Ethernet problems, 4-44

## M

---

### M9060-A load module, 2-11

### Maintenance mode, 3-20

- address specifiers, 3-37
- command keywords, 3-43
- command qualifiers, 3-41
- command syntax, 3-37

### Maintenance mode (Cont.)

- console commands, 3-45
- description of, 3-36
- special characters for, 3-36
- symbolic addresses, 3-38

### Maintenance mode commands

- CONFIGURE, 2-2

### Mass storage

- See DSSI

### MDM operating system

- and restart procedures, 3-14
- boot devices, supported, 3-12
- bootstrap, 3-10
- supported boot flags, 3-12

### MEMCSR 0-15, 4-32

### Memory

- acceptance testing of, 4-32
- cache, on KN210 CPU, 1-8
- controller chip (CMCTL), on KN210 CPU, 1-9
- isolating FRU, 4-32, 4-40
- maximum supported, in KN210 systems, 1-9
- MS650-BA, description of, 1-15
- testing, 4-40

### Messages

- console error, 4-28
- system halt, 4-27
- VMB error, 4-29

### Module

- configuration, 2-2
- order, in backplane, 2-1
- self-tests, 4-45

### MOVE command, 3-59

- MS650-BA memory module, description of, 1-15

- MS650-BF option kit, contents of, 1-15

## N

---

### Network interface

- on KN210 I/O module, 1-11

### NEXT command, 3-61

### Node name



## Node name (Cont.)

DSSI, changing, 2-5

## Normal mode

command syntax in, 3-18

console commands for, 3-17

conventions used for description  
purposes, 3-19

description of, 3-15

environmental variables for, 3-16

special characters for, 3-15

## O

---

### OCP, 4-47

cabling, 2-8

### Operating system bootstrap

and bootstrap support, 3-8

and ULTRIX-32 procedures, 3-9

conditions for, 3-8

of MDM, description of, 3-10

of ULTRIX-32, 3-9

### Operating system restart, MDM, 3-14

### Operating system support, 1-1

### Operation switch

and action position at power-up,  
3-6, 3-7

and maintenance position at  
power-up, 3-6

and normal position at power-up,  
3-5

### Operation switch, on H3602-AB, 1-13

### Operator console panel

See OCP

## P

---

### Panel, CPU I/O

See H3602-AB CPU I/O panel

### Parameters

for diagnostic tests, 4-6

in error display, 4-17

### PARAMS local program, 4-36, 4-52 commands, 4-52

Physical address locations, and  
accessing through R3000  
processor, B-1

### Physical memory

symbolic addresses for, 3-39

Power-up sequence, 3-5

Power values, 2-12

printenv command, in normal mode,  
3-33

## Q

---

### Q22-bus

interface chip (CQBIC), 1-10

## R

---

R3000 RISC chip, on KN210 CPU,  
1-8

REPEAT command, 3-63

### RF30 local programs

DRVEXR, 4-36

DRVST, 4-36

HISTORY, 4-36

PARAMS, 4-36

### RF71 ISE

cabling, 2-8

node ID switches, 2-4

### RF-series ISE, local programs

DRVEXR, 4-49

DRVST, 4-48

ERASE, 4-51

HISTORY, 4-50

list of, 4-47

PARAMS, 4-52

### RF-series ISEs

configuration errors, 4-47

diagnostic error codes, 4-55

diagnostics, 4-46

RISC chip, R3000, on KN210 CPU,  
1-8

### ROM-based diagnostics

and memory testing, 4-40

## S

---

### Scripts

- calling sequence for, 4-9
- commonly used, 4-9
- creating with 9E utility, 4-11
- description of, 4-7
- list of, 4-8

SEARCH command, 3-64

Self-test, for modules, 4-45

SET command, 3-66

setenv command, in normal mode,  
3-34

SET HOST/DUP command, 3-66

SHOW command, 3-69

SII chip, on KN210 I/O module,  
1-11

START command, 3-73

### Switch

- function, on H3602-AB, 1-13
- function, set to test position at  
power-up, 3-6
- operation, on H3602-AB, 1-13
- operation, set to action position at  
power-up, 3-6, 3-7
- operation, set to maintenance  
position at power-up, 3-6
- operation, set to normal position  
at power-up, 3-5

Symbolic addresses, 3-38

for any address space, 3-41

for GPRs, 3-38

for IPRs, 3-38

for physical memory, 3-39

Systems, for KN210 CPU module  
set, 1-5

## T

---

TEST command, 3-74

### Tests, diagnostic

- See also* Troubleshooting,  
Loopback
- list of, 4-4
- parameters for, 4-6

Time-of-year clock, on KN210 CPU,  
1-9

Timers, on KN210 CPU, 1-9

Troubleshooting, 4-37

memory failures, 4-40

memory failures, additional  
suggestions for, 4-43

with FE utility, 4-37

## U

---

ULTRIX-32 operating system

boot devices, supported, 3-9

bootstrap, 3-9

bootstrap procedure for, 3-9

Exerciser and Verf commands,  
list of, A-1

Unit number

DSSI, changing, 2-6

UNJAM command, 3-75

unsetenv command, in normal mode,  
3-35

Utilities, diagnostic, 4-4

## V

---

Virtual memory bootstrap

*See* VMB

VMB, 3-11

boot flags, 3-12

error messages, 4-29

## X

---

X command (binary load and  
unload), 3-76